Faculty of Computer Science and Information Technology

UMTS Core Network Simulator

Supervisor : **Mr. Ang Tang Fong**
Moderator : **Dr. Rosli Salleh**
Name : **Too Sea Chen**
Matric Number : **WET 000285**

Abstract

The wireless industry continues to offer new opportunities and challenges: The rise of wireless devices, plus the rising bandwidth requirements envisioned the potential packet data applications, is creating a huge collection of attempt to shore up that goal. IMT2000 specification has some platforms all called 3G, and since they meet up particular data throughput necessities, the decisions are enormous and vital for an operator to put together.

The UMTS development has introduces an all-IP multimedia network architecture. This step in the evolution represents a transform in the general call model. Particularly, both voice and data are mostly handled in the same method all the way from user terminal to the final destination. This architecture can be considered the essential convergence of voice and data.

The aim of this thesis report is to develop a UMTS Core Network simulator based on 3GPP Release 5 All-IP Network Architecture. It can offer a means for researches and network planners to analyze the behavior of UMTS Core Network without the expense of building a real network. An main feature of this simulator is multithreading, that support concurrent processing. More than one process can be executed at the same time. Hence, an object-oriented programming approach had been choose to implement it.

Acknowledgement

I would like to take this opportunity to thank those people who had helped me throughout this assignment.

First of all, I would like to express my gratitude to my respected supervisor, Mr. Ang Tan Fong for his support, valuable guidance, encouragement and constructive comments during this project.

Special thanks to Dr. Rosli Salleh for being a considerate and kind moderator who has contributed suggestions and ideas in this project.

Last but not least, my deepest appreciation to my friends, Chai Cheak Luai, Teo Mei Ling and Ng Sook Kei for their continuous support to complete this project.

Table of Content

List of Figures

List of Table

N-PDU	Network Protocol Data Unit
NS	Network Service
NSAPI	Network layer Service Access Point Identifier
NSS	Network SubSystem
ODB	Operator Determined Barring
P-TMSI	Packet TMSI
PCU	Packet Control Unit
PDCH	Packet Data CHannel
PDCP	Packet Data Convergence Protocol
PDN	Packet Data Network
PDP	Packet Data Protocol, e.g. IP
PDU	Protocol Data Unit
PMM	Packet Mobility Management
PPF	Paging Proceed Flag
PPP	Point-to-Point Protocol
PTP	Point To Point
PVC	Permanent Virtual Circuit
RA	Routeing Area
RAB	Radio Access Bearer
RAC	Routeing Area Code
RAI	Routeing Area Identity
RANAP	Radio Access Network Application Protocol
RAU	Routeing Area Update
RLC	Radio Link Control
RNC	Radio Network Controller
RNS	Radio Network Subsystem
RNTI	Radio Network Temporary Identity
RRC	Radio Resource Control
SBSC	Serving Base Station Controller
SBSS	Serving BSS
SGSN	Serving GPRS Support Node
SM	Short Message
SM-SC	Short Message service Service Centre
SMS-GMSC	Short Message Service Gateway MSC
SMS-IWMSC	Short Message Service Interworking MSC
SN-PDU	SNDCP PDU
SNDC	SubNetwork Dependent Convergence
SNDCP	SubNetwork Dependent Convergence Protocol
SPI	Security Parameter Index
SRNC	Serving RNC
SRNS	Serving RNS
TCAP	Transaction Capabilities Application Part
TCP	Transmission Control Protocol
TFT	Traffic Flow Template
TEID	Tunnel Endpoint IDentifier
TLLI	Temporary Logical Link Identity
TOM	Tunnelling Of Messages
TOS	Type of Service
TRAU	Transcoder and Rate Adaptor Unit
UDP	User Datagram Protocol
UEA	UMTS Encryption Algorithm
UIA	UMTS Integrity Algorithm
URA	UTRAN Registration Area
USIM	User Service Identity Module
UTRAN	UMTS Terrestrial Radio Access Network

Currently, voice has been the main wireless application with the use of the short message service (SMS) being the biggest packet data service.

1.2 Introduction to Universal Mobile Telecommunications Service (UMTS)

Universal Mobile Telecommunications Service (UMTS) represents a development of Global System for Mobile communications (GSM) to carry third-generation (3G) capabilities. The UMTS system can be separated into two major segments: the access network, called the UMTS Terrestrial Radio Access Network (UTRAN) and the switching and routing infrastructure, or UMTS Core Network (CN).

1.2.1 UMTS Core Network (CN)

The UMTS Core Network (CN) rationally separated into a Circuit Switched (CS) domain, a Packet Switched (PS) domain and an IP Multimedia (IM) subsystem. [5]

1.2.1.1 Circuit Switched (CS) domain

The Circuit Switched (CS) domain refers to the set of all the CN entities offer "CS type of connection" for user traffic as well as all the entities sustaining the related signaling. A "CS type of connection" is a link for which dedicated network resources are allocated at the connection establishment and free at the connection

release. The entities definite to the CS domain are: Mobile-services Switching Centre (MSC), Gateway MSC (GMSC) *and* Visitor Location Register (VLR).

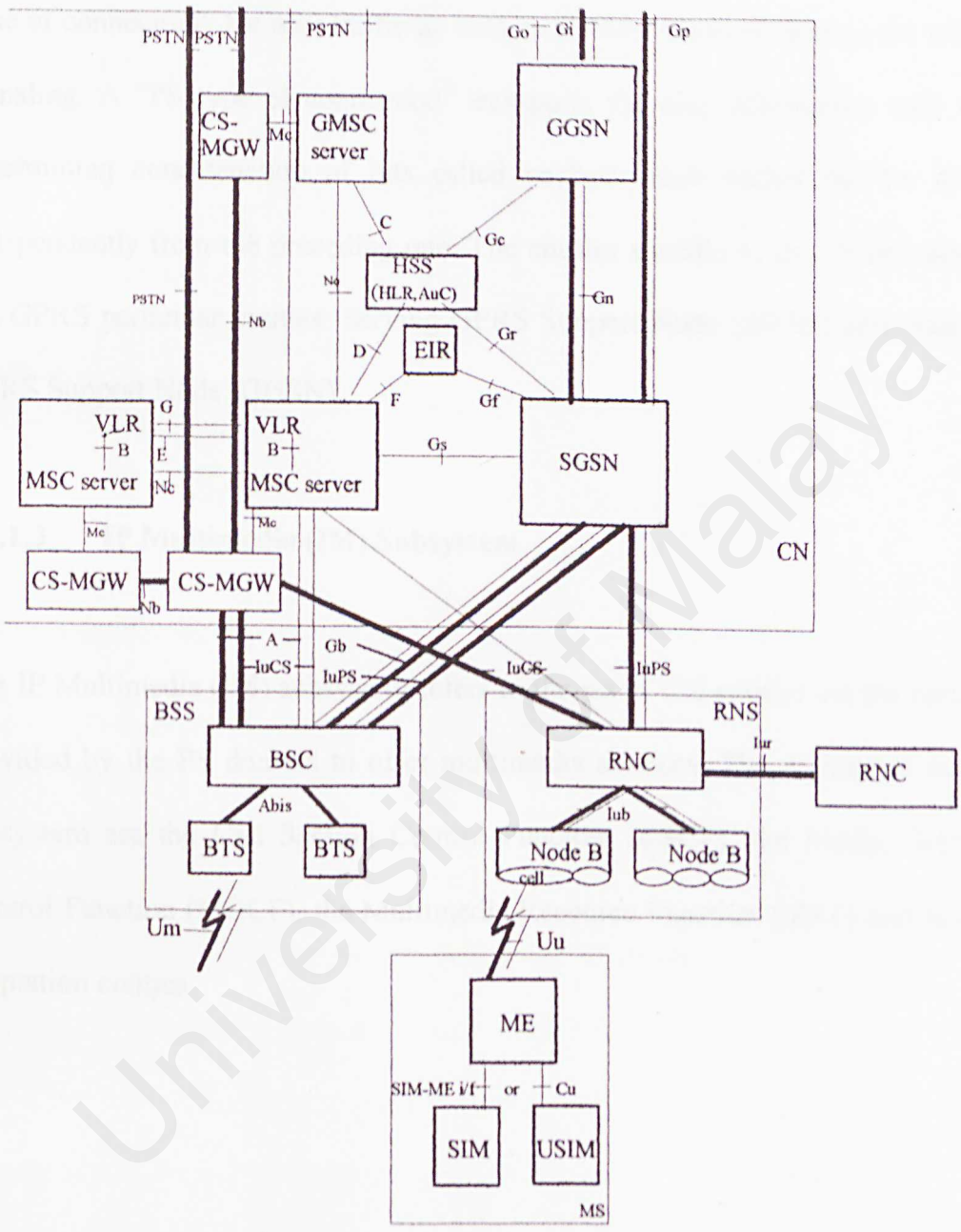


Figure 1-2: Configuration of a PLMN supporting CS and PS services and interfaces

1.2.1.2 Packet Switched (PS) domain

The Packet Switched (PS) domain refers to the set of all the CN entities offer "PS type of connection" for user traffic as well as all the entities sustaining the related signaling. A "PS type of connection" transports the user information with self-determining concatenation of bits called packets: each packet can be routed independently from the preceding one. The entities specific to the PS domain are the GPRS particular entities: Serving GPRS Support Node (SGSN) and Gateway GPRS Support Node (GGSN).

1.2.1.3 IP Multimedia (IM) Subsystem

The IP Multimedia (IM) subsystem refers to the set of CN entities via the services provided by the PS domain to offer multimedia services. The entities of the IM subsystem are the Call Session Control Function (CSCF), the Media Gateway Control Function (MGCF), the Multimedia Resource Function (MRF) and several adaptation entities.

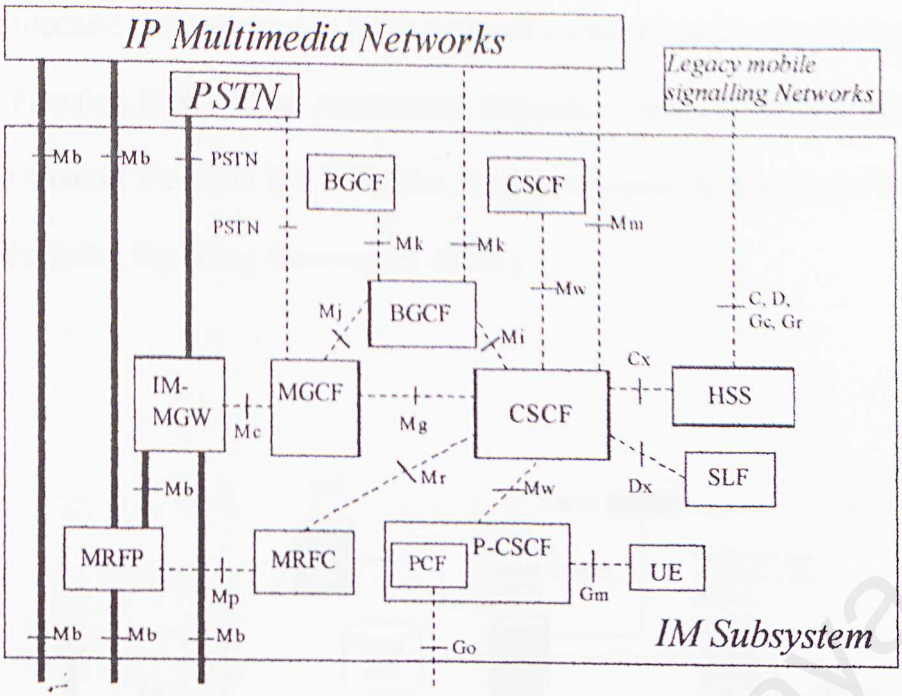


Figure 1-3: Configuration of IM Subsystem entities

1.3 Introduction to 3GPP UMTS Release 5 All-IP Network Architecture

The hottest UMTS development is the introduction of all-IP multimedia network architecture. This step in the progression represents a transform in the on the whole call model. Particularly, both voice and data are mainly handled in the same way all the way from the user terminal to the final destination. This architecture can be considered the crucial convergence of voice and data.

From this architecture, voice and data no longer require split interfaces like UMTS Release 1999 or Release 4. Just a single Iu interface can transmit all the media. Within the core network, that interface terminates at the SGSN and there is no break up media gateway.

This architecture has a amount of latest network elements, especially the Call State Control Function (CSCF), the Multimedia Resource Function (MRF), the Media Gateway Control Function (MGCF), the Transport Signaling Gateway (T-SGW), and the Roaming Signaling Gateway (R-SGW).

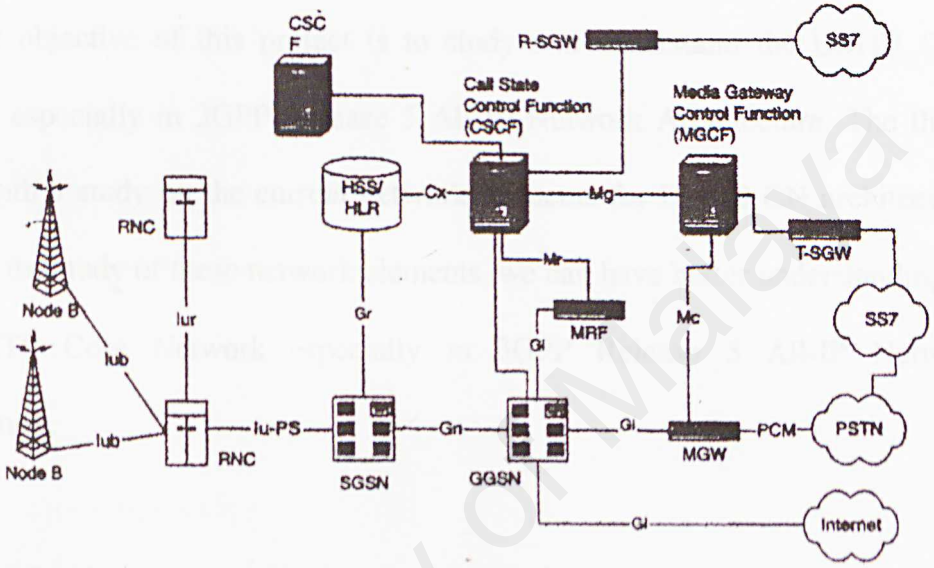


Figure 1-4: 3GPP Release 5 IP Multimedia Network Architecture

A main aspect of all-IP architecture is the fact that the user equipment is really improved. Important logic is placed inside the UE. In reality, the UE supports the Session Initiation Protocol (SIP). The UE efficiently becomes a SIP user agent. As such, the UE has far better control of services than before.

It should be noted that the Release 5 all-IP architecture is an improvement to an existing Release 1999 or Release 4 network. It is successfully the addition of a new domain in the core network, the IP Multimedia (IM) subsystem. This new domain, which enables both voice and data to be passed over IP all the way from the phone,

uses the services of the PS domain for transport purposes. That is, it uses the SGSN, GGSN, Gn and Gi which these nodes and interfaces that belong to the PS domain.

1.4 Project Objectives

The first objective of this project is to study and understand the UMTS Core Network especially in 3GPP Release 5 All-IP Network Architecture. The thesis begins with a study on the current network elements for UMTS CN architecture. Through the study of these network elements, we can have better understanding on the UMTS Core Network especially in 3GPP Release 5 All-IP Network Architecture.

The second objective is to develop the UMTS Core Network Simulator base on 3GPP Release 5 All-IP Network Architecture. The simulator is developed using an object-oriented approach to take advantage of the features such as modularity, extensibility, reusability and others. Finally is to create a portable and user-friendly simulator.

1.5 Project Scope

The main scope of this thesis is to create components needed by the UMTS Core Network simulator to properly simulate the 3GPP Release 5 All-IP Network Architecture. The simulator components needed for an UMTS Core Network, such

as GGSN, SGSN, CFCS, MGCS and physical link in order to get the idea of how the UMTS Core Network simulator built.

This project will develop an UMTS Core Network simulator that will have the following features:

- Platform independent
- Default parameters for simulator components
- A user friendly graphical user interface (GUI)
- Integrated data analysis tool or mechanisms recording the simulator results and network configuration
- Discrete event schedule

1.6 Project Schedule

Project Stage	June				July				Aug				Sept				Oct				Nov				Dec				Jan
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1
Preliminary Study and Planning																													
Literature Study																													
System Analysis																													
System Design																													
Research																													
Development and Coding																													
Unit Testing																													
System Testing																													
Documentation																													
Implementation and Maintenance																													

Figure 1-5: Project Schedule

1.7 Report Organization

Chapter 1, "Introduction"

This chapter gives a brief overview of 3G and UMTS architecture. Project objective, project scope and project schedule are described here.

Chapter 2, "Literature Review"

Survey on different UMTS technologies is covered in this chapter. This chapter contains programming techniques, languages and tools that will be used in assisting the development of the system. Besides, survey on UMTS CN 3GPP Release 5 All-IP Network Architecture is done. Each network elements is explained in detailed.

Chapter 3, "System Analysis"

The details of UM JaNetSim Simulator and network elements functions are highlighted in this chapter. In this chapter, the considerations taken into account in discussing programming language choice and selected tools. This chapter also emphasis on functionality and non-functionality features of the simulator.

Chapter 4, "System Design"

Chapter 4 concentrates in the design of simulation tool. In this chapter, an overall architecture will be shown, followed by class design, program flow design and simulator design overview.

Chapter 5, "Implementation"

Chapter 5 outlines the modules, which had been planned out in the design phase. The classes being implemented are discussed further in this chapter.

Chapter 6, "Testing"

Chapter 6 consists of simulator testing. This section describes component testing, module testing and system testing. Finally in the chapter 6, the conclusion on this project is made accordingly.

Chapter 2 – Literature Review

This chapter is to study environment of 3GPP UMTS Release 5 All-IP Network Architecture. Besides, the reason of this view is to get better understanding on the development tools that can be used to build up a network simulator.

Apart of that, this chapter also reviews with brief foreword of the current network simulator. A detail explanation for the simulators that includes their advantages and disadvantages is provided. It will give an idea how to get better the limitation and accomplish the requirements needed.

2.1 3GPP UMTS Release 5 All-IP Network Components

The following are the studies, which have been taken during this project:

- Serving GPRS Support Node (SGSN)
- Gateway GPRS Support Node (GGSN)
- Proxy Call Session Control Function (CSCF)
- Interrogating-CSCF
- Serving-CSCF

2.1.1 Serving GPRS Support Node (SGSN)

The location register function in the SGSN stores two types of subscriber data needed to handle originating and terminating packet data transfer: [5]

- Subscription information:
    - The IMSI;
    - One or more temporary identities;
    - Zero or more PDP addresses.
- Location information:
    - Depending on the operating mode of the MS, the cell or the routeing area where the MS is registered;
    - The VLR number of the associated VLR (if the Gs interface is implemented);
    - The GGSN address of each GGSN for which an active PDP context exists.

2.1.2 Gateway GPRS Support Node (GGSN)

The location register function in the GGSN stores subscriber data received from the HLR and the SGSN. There are two types of subscriber data needed to handle originating and terminating packet data transfer:

- Subscription information:

- The IMSI;
- Zero or more PDP addresses.

- Location information:
  - The SGSN address for the SGSN where the MS is registered.

2.1.3 Proxy Call Session Control Function (CSCF)

The Proxy-CSCF (P-CSCF) [3] is the first contact point within the IM CN subsystem. Its address is discovered by UEs following PDP context activation. The P-CSCF behaves like a Proxy, it accepts requests and services them internally or forwards them on. The P-CSCF shall not modify the Request URI in the SIP INVITE message. The P-CSCF may behave as a User Agent (as defined in the RFC 3261 [12] or subsequent versions), i.e. in abnormal conditions it may terminate and independently generate SIP transactions.

The Policy Control Function (PCF) is a logical entity of the P-CSCF. If the PCF is implemented in a separate physical node, the interface between the PCF and the P-CSCF is not standardized.

The functions performed by the P-CSCF are:

- Forward the SIP register request received from the UE to an I-CSCF determined using the home domain name, as provided by the UE.

- Forward SIP messages received from the UE to the SIP server (e.g. S-CSCF) whose name the P-CSCF has received as a result of the registration procedure.

- Forward the SIP request or response to the UE.

Detect and handle an emergency session establishment request as per error handling procedures defined by stage-3.

- Generation of CDRs.

- Maintain a Security Association between itself and each UE.

- Should perform SIP message compression/decompression.

- Authorisation of bearer resources and QoS management

2.1.4 Interrogating-CSCF

Interrogating-CSCF (I-CSCF) is the contact point within an operator's network for all connections destined to a subscriber of that network operator, or a roaming subscriber currently located within that network operator's service area. There may be multiple I-CSCFs within an operator's network. The functions performed by the I-CSCF are:

- Registration
    - Assigning a S-CSCF to a user performing SIP registration Session-related and session-unrelated flows.

- Route a SIP request received from another network towards the S-CSCF.
- Obtain from HSS the Address of the S-CSCF.
- Forward the SIP request or response to the S-CSCF determined by the step above

- Charging and resource utilisation:
  - Generation of CDRs.

2.1.5 Serving-CSCF

The Serving-CSCF (S-CSCF) performs the session control services for the UE. It maintains a session state as needed by the network operator for support of the services. Within an operator's network, different S-CSCFs may have different functionalities. The functions performed by the S-CSCF during a session are:

- Registration
  - May behave as a Registrar or subsequent versions, i.e. it accepts registration requests and makes its information available through the location server (eg. HSS).

- Session-related and session-unrelated flows
  - Session control for the registered endpoint's sessions.
  - May behave as a Proxy Server and it accepts requests and services them internally or forwards them on, possibly after translation.

- May behave as a User Agent and it may terminate and independently generate SIP transactions.
- Interaction with Services Platforms for the support of Services
- Provide endpoints with service event related information (e.g. notification of tones/announcement together with location of additional media resources, billing notification)
- On behalf of an originating endpoint (i.e. the originating subscriber/UE)
    - Obtain from a database the Address of the I-CSCF for the network operator serving the destination subscriber from the destination name of the terminating subscriber (e.g. dialled phone number or SIP URL), when the destination subscriber is a customer of a different network operator, and forward the SIP request or response to that I-CSCF.
    - When the destination name of the terminating subscriber (e.g. dialled phone number or SIP URL), and the destination subscriber is a customer of the same network operator, forward the SIP request or response to an I-CSCF within the operator's network.
    - Depending on operator policy, forward the SIP request or response to another SIP server located within an ISP domain outside of the IM CN subsystem.

- Forward the SIP request or response to a BGCF for call routing to the PSTN or CS Domain.

- On behalf of a destination endpoint (i.e. the terminating subscriber/UE)

  - Forward the SIP request or response to a P-CSCF for a MT procedure to a home subscriber within the home network, or for a subscriber roaming within a visited network where the home network operator has chosen not to have an I-CSCF in the path

  - Forward the SIP request or response to an I-CSCF for a MT procedure for a roaming subscriber within a visited network where the home network operator has chosen to have an I-CSCF in the path.

  - Modify the SIP request for routing an incoming session to CS domain according to HSS and service control interactions, in case the subscriber is to receive the incoming session via the CS domain.

  - Forward the SIP request or response to a BGCF for call routing to the PSTN or the CS domain.

- Charging and resource utilisation:

  - Generation of CDRs.

2.2 Introduction of various Network Simulators

There are presently a few simulators on the market. The following sub-sections explain different simulators that have been implemented, including REAL Network Simulator, NIST ATM/HFC and INSANE (Internet Simulated ATM Networking Environment).

2.2.1 REAL Network Simulator

The Real network simulator [9] is a network simulator planned for testing congestion and flow control mechanisms. Real is a simulator for studying the dynamic performance of flow and congestion control schemes in packet switch data networks. It provides users with a way of specifying such networks and observes their activities.

The simulator takes as input a scenario, which is a explanation of network topology, protocols, workload and control parameters. It produces as output statistics such as the amount of packets sent by each source of data, the queuing delay at each point, the number of dropped and retransmitted packets and other similar information. REAL, as distributed, run on Sun3s, Sparcs, MIPS boxes, Vaxen and 3B2, under 4.3BSD-like operating systems: SunOS, IRIX, UMIPS, Ultrix etc.

Advantage

- User can adjust the simulator software to accommodate network components.
- Flexible.

Disadvantages

- No graphical user interface (GUI) representation capabilities.
- Not a cross-platform simulator.
- Not object-oriented.
- User must have strong knowledge in C programming language.

2.2.2 NIST ATM/HFC Network Simulator

This NIST ATM/HFC Network Simulator [11] was developed at the National Institute of Standards and Technology (NIST) to offer a flexible testbed for studying and evaluating the performance of ATM and HFC network without the expense of building a real network. This simulator is written in C Language whereby it is written in structural programming approach. Normally, the simulator is tools that give user an interactive modelling environment with a graphical user interface which provides the user with a means to show the topology of the network, define the parameters and connectivity of the network, log data from simulation run and load the network configuration.

Advantages

- Allows user to change the parameters of each component's operation, measure network activity, save/load different simulation configuration and log data during simulation execution.
- Allows user to form different network topologies.
- Provides graphical user interface.
- Provides various instantaneous performance measurement displayed in graphical/ text form on the screen while the simulation is running.

Disadvantages

- User or programmers need to have strong foundation in C programming language to customize and know the simulator's components. Besides, it is using procedural approach whereby the components have overlapped functions between the components. This is not supposed to happen in object-oriented programming approach.
- User might faces trouble setting up the network topology because of the requirement to consider a large number of parameters.
- Not object-oriented and not a cross-platform simulation, that is the reason why it is not widely used. The simulator only can run in UNIX or LINUX platform.

2.2.3 INSANE Network Simulator

INSANE (Internet Simulated ATM Networking Environment) [8] is a network simulator intended to test various IP-over ATM algorithms with realistic traffic loads derived from empirical traffic measurements. INSANE's ATM protocol stack provides real-time guarantees to ATM virtual circuits by using Rate Controlled Static Priority (RCSP) queuing. ATM signalling is performed using a protocol alike to the Real-Time Channel Administration Protocol (RCAP).

Internet protocols supported include large subset of IP, TCP and UDP. In particular, the simulated TCP implementation performs connection management, slow start, flow and congestion control, retransmission and fast retransmit. Various application simulators mimic the behaviour of standard Internet applications to provide a realistic workload, including: telnet, ftp, WWW, real-time audio and real-time video. INSANE is designed to run big simulations whose results are processed off-line.

Advantages

- Written in C++ language, so it is an object-oriented programming approach.
- It works quite well on distributed computing clusters (although simulations are all sequential processes, a large number of them can easily be run in parallel).

Disadvantages

- Output performance only can be viewed in text based.
- Not a very user friendly environment.
- No Graphical User Interface (GUI).

2.2.4 NetSim (Network Simulator)

NetSim (Network Simulator) is a computer program that simulates theoretical detection and location capabilities of seismic networks. This simulation is used to assess seismic network capabilities to identify and locate explosions and earthquakes for monitoring of test ban limitation treaties. NetSim is derived from earlier program such as NETWORTH and SNAP/D. The current version of NetSim capable of simulating the detection of regional Pn, PG, Sn and Lg as well as teleseismic P, PKP and S. Location assessment may be performed with all these phases. Furthermore, NetSim attempts to identify the frequency at which the detection will have the largest signal to noise ratio (SNR).

Advantages

- Can be used as a component of a simulation testbed capable of simulating a full parallel system.
- Suitable in simulating large networks.

Disadvantages

- User must have a strong knowledge in C programming language.

- Does not give user an interactive graphical user interface (GUI) environment.
- No other traffic management and QoS are considered to be present.
- This simulator has not been thoroughly tested and may contain bugs.

2.2.5 JaNetSim (Java Network Simulator)

JaNetSim simulator [12] is a flexible testbed for studying and evaluating the performance of ATM network without the expense of building a real network. This simulator is written in Java Language whereby it is written in object-oriented programming approach. Actually, this simulator is a Java version of NIST ATM/HFC Network Simulator enhanced with object-oriented features. Typically, the simulator is a tool that gives user an interactive modelling environment with a graphical user interface which provides the user with a means to display the topology of the network, define the parameters and connectivity of the network, log data from simulation run and save and load the network configuration.

Advantages

- Allows user to create different network topologies.
- Provide a very user friendly environment.
- Better graphical user interface.
- Allows user to adjust the parameters of each component's operation, measure network activity, save/load different simulation configuration and log data during simulation execution.

- Output performance can be viewed in text based and graphical representation on the screen whiles the simulation running.
- Cross-platform simulator.

Disadvantages

- Required a lot of memory processing during the simulation.
- Written in Java application, so it is not webs enable approach.

2.2.6 Comparison

The following table give a comparison among the studied simulators. Features being compared are discrete event simulation, object-oriented, graphical user interface (GUI), multithread and web-enable.

Table 2-1: Comparison among Various Simulators

Simulator	Discrete event simulation	Object-oriented	GUI	Multithread	Web-enable	Platform independent
NIST ATM/HFC	√	X	√	X	X	X
INSANE	√	√	Poor	X	X	X
REAL	√	X	Poor	X	X	X
NETSIM	√	X	Poor	X	X	X
JANETSIM	√	√	√	√	X	√

The entire simulator is discrete event simulators. However, OPNET, OMNet++, PARSEC and Delsi were developed using object-oriented approach. On the other hand, NIST ATM/HFC, OMNet++ and OPNET provide a powerful graphical user interface while others are poor in user interface design. Unfortunately, only PARSEC supports multithreading and none of the simulators are web-enable and platform independent.

2.3 JaNetSim Simulator

2.3.1 JavaSim

The JavaSim object is the main object of the simulator. It keeps a list of all the network components (all are descendents of SimComponent), and a list (a queue) of all events (in the form of SimEvent). Every component contains a set of parameters (all inherit from SimParameter).

2.3.2 SimComponent

This is the most important class to understand in the simulator in order to develop new components. Every network component in the simulation must inherit SimComponent. The SimComponent class itself should not be instantiated (although this is possible) because it only provides the skeleton for an actual component. A new component should extends SimComponent and override its various methods in order to provide meaningful operations for the component.

2.3.3 SimClock

The simulator is event driven. Components send each other events in order to communicate and send cells through the network. The software contains an event manager, which provides a general facility to schedule and send, or "fire" an event.

An event queue is maintained in which events are kept sorted by time. To fire an event, the first event in queue is removed, the global time is set to the time of that event and any action scheduled to take place is undertaken.

Events can be scheduled at the current time or at any time in the future. Scheduling events for the past is considered illogical. Events scheduled at the same time are not guaranteed to fire in any particular order. Simulator time is maintained by the event manager in units of ticks. The time is maintained as an unsigned 32-bit value. The simulator time represented by one tick can be changed by software modification, but not by the simulator user. It provides a set of time translation function (all static) for normal translation between tick and actual time (microseconds, seconds etc.).

2.3.4 SimParameter

Every SimComponent can have internal parameters (not shown/accessible by users) or external parameters (shown/accessible by users). All external parameters must inherit SimParameter. By extending SimParameter, one obtains parameter logging and meter display features automatically. *SimParamInt, SimParamDouble, SimParamBool and SimParamString.* Obviously, these 4 objects provide support for integer, double, Boolean and string parameters. Other types of parameters can be created by extending SimParameter accordingly.

2.3.5 SimEvent

Every SimComponent communicates with each other by enqueuing SimEvent for the target component. For example, when component X wants to send a packet to component Y, component X creates a SimEvent that specifies Y as its destination and enqueue the event. The SimEvent object also contains a time so that this event is fired at exactly the specified time. Component Y will then be able to react to the event accordingly.

2.3.6 Link Components

This component simulates the physical medium (copper wire or optical fiber) on which cell is transmitted. The user may choose the link speed from a list of several different standard rates. The user also specifies the length of the link. The output parameter reported by the simulator is link utilization in terms of bit rate (Mbits/s).

- Modularity – Each object forms a separate entity whose internal workings are decoupled from other parts of the system. [14]
- Maintainability – Maintenance and modification of objects can be done individually.
- Simplicity – It is simple and less complex using OOP approach while building programs, which attempts to model the objects interaction of the real world. Any changes are easy to modify with no much affect within the entire system.

2.5 Programming Tools

Java language, C++ language are some of the examples that implement this object-oriented concept. However, in order to full fill the project objective requirement, Java language had been chosen because this language supports multithreading, platform and browser independent.

2.5.1 Java Programming Language

There are a few types of Java programming tools available in market. There are Java 2 SDK Standard Edition, Borland JBuilder and Microsoft Visual J#.

2.5.1.1 Java 2 SDK, Standard Edition

The latest version of this software is 1.3.1. Below are some of the features that available in this software and also some enhancement [10] made from previous version:

- Applet Deployment Enhancements
- Swing Enhancements
- Security Enhancements
- Java 2DTM Enhancements
- AWT Enhancements
- Networking Enhancements
- Java Sound
- Accessibility

Performance

Many enhancements have been made to improve the performance of the Java 2 SDK and Java 2 Runtime Environment. These changes include the addition of the Java HotSpot™ Client Virtual Machine. The Solaris and Linux versions of the Java 2 SDK and Java 2 Runtime Environment also contain the Java HotSpot Server VM. Both the Client and Server VMs contain state-of-the-art Java HotSpot technology. The Client VM is tuned to maximize performance on client systems, improving performance in areas of start-up time and memory footprint. The Server VM is tuned to maximize performance of program execution speed and is aimed at server application that is less concerned with start-up and memory footprint.

2.5.1.2 Borland JBuilder™ 7 Enterprise

JBuilder 7 Enterprise [7] makes EJB, ™ Web, XML, and database application development easier with two-way visual designers and rapid deployment to leading J2EE™ platform application servers, including BEA® WebLogic,® IBM® WebSphere, ® iPlanet, ™ Oracle9i, ® and the integrated Borland® Enterprise Server. Enhance developer productivity and take advantage of extreme programming with UML™ code visualization, refactoring, and unit testing. Develop and deploy applications on the Windows, ® Linux, ® Solaris, ™ and Mac® OS platforms. Integrate with enterprise build processes using Apache™ Ant. Efficiently collaborate in teams with support for leading version control systems.

JBuilder™ 7 Enterprise features:

- Graphical debugging
- Visual designers
- Automated wizards
- Enterprise Java Development
- AppBrowser™ integrated development environment
- Extensible source code editor
- Industry-standard database connectivity

2.5.1.3 Microsoft Visual J#

Microsoft Visual J#™ .NET [6] is a development tool for Java-language developers who want to build applications and services on the Microsoft .NET Framework. Visual J# .NET joins more than 20 previously announced languages with its ability to target the .NET Framework and first-class XML Web services.

Visual J# .NET provides:

- The easiest transition for Java-language developers into the world of XML Web services.
- Dramatically improved interoperability of Java-language programs with existing software written in a variety of other programming languages.
- The opportunity for Microsoft Visual J++® customers and other Java-language programmers to take advantage of existing investments in skills and code while fully utilizing the Microsoft platform today and into the future.

Visual J# .NET includes technology that enables customers to migrate Java-language investments to the .NET Framework. Existing applications developed with Visual J++ can be easily modified to execute on the .NET Framework, interoperate with other .NET-connected languages and applications, and incorporate new .NET functionality such as Microsoft ASP.NET, Microsoft ADO.NET, and Microsoft Windows® Forms. Further, developers can use Visual J# .NET to create entirely new .NET-connected applications.

Visual J# .NET provides:

- Full integration with Visual Studio .NET. Visual J# .NET provides programming tools support through its integration with the award-winning Visual Studio .NET integrated development environment (IDE). All the features of the IDE are easily accessible to the Visual J# developer.
- Full integration with the .NET Framework. Visual J# .NET is designed to take full advantage of the .NET Framework, including ASP.NET, ADO.NET, Windows Forms, and XML Web services, as well as full cross-language integration.
- Visual J++ 6.0 upgrades tools. Visual J# .NET includes tools to automatically upgrade and convert existing Visual J++ 6.0 projects and solutions to the new Visual Studio .NET format. These tools ensure that an existing Visual J++ 6.0 developer can move easily to Visual J# .NET and produce .NET-based applications and components.

2.6 Summary

3GPP UMTS Release 5 All-IP Network Components already discuss in this chapter. Comparison of features between 5 current network simulators show that those simulators do not support multithreading, web-enable and only operate on certain platform like UNIX and LINUX. The following chapter will discuss more on programming language and programming tool choice.

Chapter 3 – System Analysis

3.1 Development Analysis

An analysis had been conceded out on the development tools to find out the most appropriate tools for the simulator. These tools include the entire platform, development software and programming language. The following are the tools used in the simulator:

Platform:

- Support multi-platforms include LINUX, UNIX and Windows.

Programming language choice:

- Object-Oriented Programming Approach due to its benefits that listed in **2.4.2** compared to procedural programming language.
- Java language. The major reason for choosing this language is because its support multithreading. This feature enables the simulator to execute a few processes concurrently at the same time. Java does not contain any additional features added like Visual J# that only can support by Microsoft software.

Java has several main features that make it an attractive programming language, including the following:

- Java supports multithreading – Java has built in support for multithreading, in which a Java program can create any number of threads that appear to execute simultaneously.
- Java is small – In designing Java, the designers deliberately left out superfluous features and cut the design to the bone. The result is that it has all the necessary features combined in an elegant and logical way. This means that the language is powerful, but easy to learn.
- Java is platform-independent – Java programs are designed to able to run on any computer without requiring changes to be made to them. This feature is almost certainly impossible with any other programming language.
- Java is object-oriented – The object-oriented approach is most popular approach to programming in the late 1990's. Furthermore, the object-oriented approach meshes well with the needs of client-server systems and distributed systems. The design of Java is completely object-oriented from the ground up. It is not a language that has object-orientedness grafted onto it as an afterthought.
- Java is secure – programs that have been transmitted over a network are checked by Java's run time system to ensure that they have not been tampered. Code produced by Java compiler is checked for validity and unauthorized actions are not allowed to perform by the program.
- Java has libraries – Because Java is a small programming language, most of its functionality is provided in the form of libraries. A whole host of libraries is available to provide many things, including Graphical User Interfaces (GUI), Internet access and other things ordinary programming languages can do.

- Java supports the Internet – one of the primary motivations of Java is to enable people to develop programs for the Internet and World Wide Web. Programs written in Java can easily be invoked using browsers like Internet Explorer and Netscape Navigator. Furthermore, Java programs can be transmitted easily around the Internet and run on any computer.
- Java is general-purpose – Although designed for writing World Wide Web applications, Java is also a truly general-purpose language. Anything that can be done in programming languages such as C++ or Ada can be done in Java.
- Java is robust – When a Java program goes wrong, it will not create damage, mayhem or uncertainty. This because the Java programs run inside a protective 'sandbox' that confines and controls the effects of any errors. Java programs are even protected against infiltration by viruses. Furthermore, garbage collection is done by Java, which prevents the Java program from corrupting memory via dangling pointer. Finally, Java objects can contain no reference to data that is external to them. This ensuring that instruction cannot contain the address of data storage in another application or in the operating system. This prevents the program from causing a crash in other programs or the operating system.

Programming tools

Borland JBuilder™ 7 Enterprise. The reasons we choose this tool because its support wide ranges of Java 2 technologies as mention in section **2.5.1.2**. Another exciting features that cannot obtain in Java SDK 1.3.1 is more user-friendly environment to help programmer when they facing programming problems. By

using Borland JBuilder™ 7 Enterprise, programming process can speed up because there are guidelines provide to help us to write any program.

3.2 JaNetSim Architecture

3.2.1 Object-Oriented Design

Figure 3-1 is the network simulator object for JaNetSim simulator. JavaSim is the main object in the simulator. Inside JavaSim, it contain many important classes include SimComponent (keeps the list of all the network simulator components), SimEvent (list of all the event occurred in the simulator). Parameter set inside every component will inherit SimParameter. All other classes are mostly helpers that provide certain services such as SimMeter, SimClock, SimLog and etc.

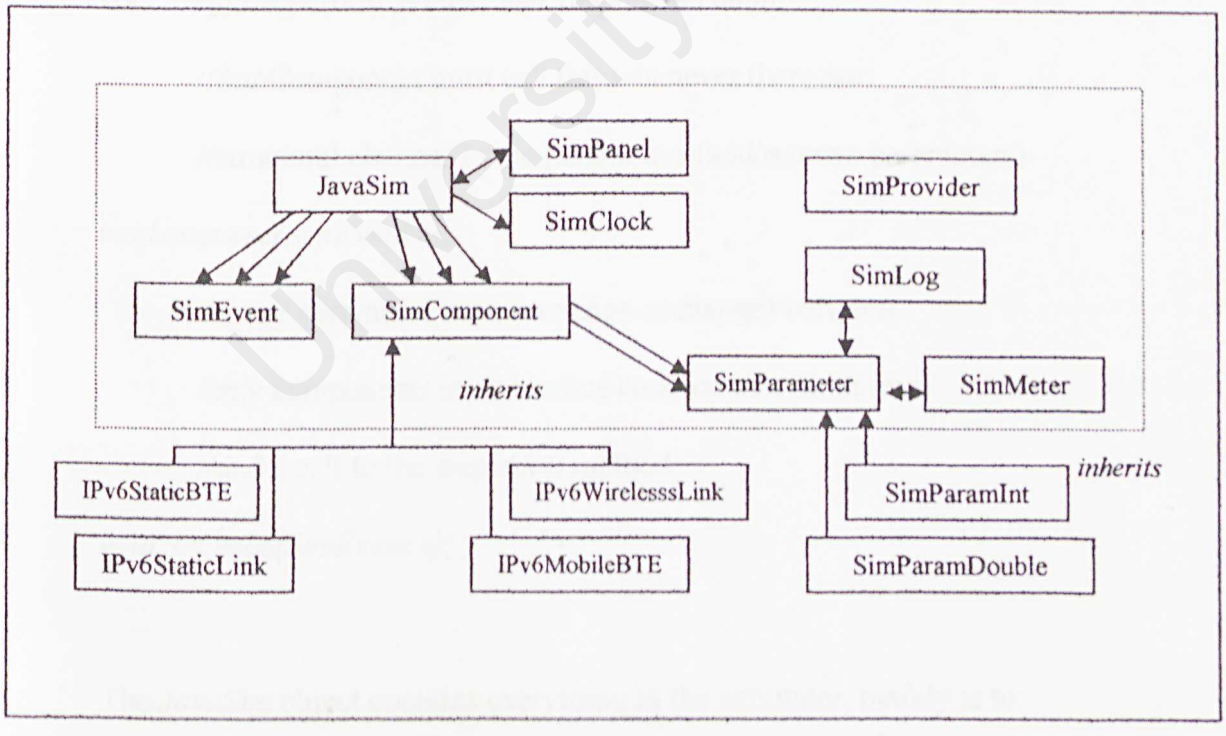


Figure 3-1: JaNetSim Simulator Objects

3.2.2 JaNetSim Class Design

This section gives a description on main engine in JaNetSim Simulator. Among the classes highlighted are *JavaSim.java*, *SimEvent.java*, *SimComponent.java*, *SimParameter.java*, *SimClock.java* and *SimProvider.java*.

JavaSim.java

long now(); //returns current simulation time (in tick)

java.util.List getSimComponents();

//returns a list of all existing SimComponent

boolean isCompNameDuplicate(String name);

//returns true if the supplied parameter *name* is

//already used by another SimComponent

void notifyPropertiesChange(SimComponent comp);

//SimComponent must call this whenever there are

//structural changes to the parameters (add/remove parameters)

void enqueue(SimEvent e);

//every communication (message exchange) between

//any components must involve creation of a SimEvent

//and a call to the *enqueue()* method

void dequeue(SimEvent e);

The *JavaSim* object contains everything in the simulator, mainly is to:

- Provide all GUI function (together with SimPanel)
- Provide the main JFrame for the application (closing it will exit)

- Provide the event manager to handle event-passing among all components

SimEvent.java

SimEvent(int aType,SimComponent src,SimComponent dest,

long aTick,Object [] params);

//the constructor needs an event type (as defined in

//SimProvider or a private event type),

//the source and destination SimComponent,a time (in ticks),

//and an array of *java.lang.Object* (which can be

//anything) holding various parameters for the event.

Upon receiving the SimEvent object, its content can be retrieved:

int getType();

//the event type

SimComponent getSource();

//the source SimComponent

SimComponent getDest();

//the destination SimComponent

long getTick();

//the time

Object [] getParams();

//the parameters

Every SimComponent communicates with each other by enqueuing SimEvent for the target component. For example, when component A wants to send a packet to

component B, component A creates a SimEvent that specifies B as its destination, and enqueue the event. The SimEvent object also contains a time so that this event fired at exactly the specified time. Component B will then be able to react to the event accordingly.

SimComponent.java

```
protected transient JavaSim theSim;
        //this is a reference to the main JavaSim object,
        //use this to obtain various services from the JavaSim
protected java.util.List neighbors;
        //this is a list of all (directly connected) neighbors
        //of the SimComponent, the content of the list should
protected java.util.List params;
        //this is a list of all parameters of a SimComponent
        //that need to be shown in the properties dialog. Its
        //content should be strictly SimParameter only!
SimComponent (String aName,int aClass,int aType,
                JavaSim aSim, Point loc);
        //the constructor, every new component MUST provide a constructor with
        //EXACTLY the above parameters and immediately call
        //super(aName,aClass,aType,aSim,loc); as the first
        //statement of the method. Any other component creation
        //operations should be handled after this call.
boolean isConnectable (SimComponent comp);
        //this is called by the simulation engine when a new
```

```
        //component is about to be connected to this component.
        //The comp is a reference to the new component.
        //The default implementation always returns true,
        //in order to provide rules for neighbor creation, override this method.
void addNeighbor (SimComponent comp);
        //this is called by the simulation engine when a new
        //neighbor is connected to this component.
        //One must override this method in order to perform
        //other neighbor creation operations, but the first
        //statement MUST be a call to super.addNeighbor(comp).
void removeNeighbor (SimComponent comp);
        //this is called by the simulation engine when a
        //neighbor is disconnected from this component.
        //One must override this method in order to perform
        //other neighbor disconnection operations, but the
        //first statement MUST be super.removeNeighbor(comp).
void removeNeighbor (java.util.List comps);
        //this is called by the simulation engine when a
        //group of neighbors is disconnected from this component.
        //One must override this method in order to perform
        //other neighbor disconnection operations. One  possible
        //implementation is to pass each item of comps to
        //the removeNeighbor(SimComponent comp) method.
Object [] compInfo (int infoid,SimComponent source,
                Object [] paramlist);
```

```
        //This method provides a way for inter-component
        //information exchange without sending runtime SimEvent.
        //This is totally component dependent and should be used
        //with care. The calling SimComponent provide an ID for the
        //information to fetch and provide any necessary parameters.
void copy (SimComponent comp);
        //This method is used to copy parameter values of another
        //SimComponent of the same type. One MUST override this
        //method in order to ensure that all necessary parameter values are copied.
void reset ();
        //This to perform a reset operation in order
        //to bring the status of the component back to the same
        //status as if it is just newly created.
void start ();
        //This to perform any operations needed when
        //the simulation starts (the user clicks the "Start" button)
void resume ();
        //This to perform any operations needed when
        //the user clicks the "Resume" button after a pause.
        //One possible use is to capture any special changes
        //that have been done by the user during the pause period.
void action (SimEvent e);
        //This is the event handler of this component and will
        //be called by the simulator engine whenever a SimEvent
        //with this component as the destination fires.
```

This is the most important class to understand in the simulator in order to development new components. Every network component in the simulation MUST inherit SimComponent. The SimComponent class itself should not be instantiated (although this is possible) because it only provides the skeleton for an actual component. A new component should extends SimComponent and override its various methods in order to provide meaningful operations for the component.

SimParameter.java

SimParameter(String aName,String compName,long creationTick,Boolean isLoggable);

//Any parameter that inherits SimParameter should provide a

//constructor that includes at least the above four

//parameters, which in turn, should be passed directly to

//super(aName,compName,creationTick,isLoggable);

//as the first statement of the constructor. The constructor

//can accept additional parameters if needed (especially

//the actual value of the parameter, which is not included

//in the basic constructor in SimParameter)

//The parameter are:

//aName – name of the parameter

//compName – name of the component the owns the parameter

//creationTick – time when the parameter is created

//isLoggable – whether the parameter can be logged in the log file

String getString();

//returns a String representation of the parameter value

// (this is used for logging purpose)

JComponent getJComponent();

//In order to create a new parameter type, one MUST also

//provide the actual graphical implementation of the

//parameter, by providing a JComponent for the simulation engine.

//Also, if the parameter allows user inputs, one must

//also create any listeners to listen to the JComponent

//in order to capture user inputs.

//The JComponent can be anything from a simple JLabel or

//JButton to a full-blown JPanel containing further child components.

//This provides very high flexibility in designing

//custom function for certain components.

SimClock.java

static double Tick2Sec(long tick); //ticks to seconds

static double Tick2MSec(long tick); //ticks to miliseconds

static double Tick2USec(long tick); //ticks to microseconds

static long Sec2Tick(double sec); //seconds to ticks

static long MSec2Tick(double sec); //miliseconds to ticks

static long USec2Tick(double sec); //microseconds to ticks

Provides a set of time translation functions (all static) for normal translation between tick and actual time (microseconds, seconds etc.).

SimProvider.java

```
Private static final String [] classes = {
        "StaticBTE",
        "MobileBTE"
        "StaticLink"
        "WirelessLink"
};
        //All the classes are defined here:
static final int EV_SELFTEST = 0;
static final int EV_RECEIVE = 1;
static final int EV_READY = 2;
static final int EV_PRIVATE = 100;
        //event type constants
final int getCompClass();
        //get the component class (e.g. staticBTE, mobileBTE, wireless link etc.)
final int getCompType();
        //get the component type (further division under a component class)
```

The SimProvider object, defines all the public events (this is the only part of the simulation engine that requires recompilation in order to allow development of new SimComponent and event types). All private events should be defined within the particular SimComponent source itself. All private events must be greater than a constant (SimProvider.EV_PRIVATE) define in SimProvider. So, the first private event should have a value of *SimProvider.EV_PRIVATE + 1*, the next *SimProvider.EV_PRIVATE + 2*, and so on.

Every SimComponent MUST has a Component class (not to be confused with the java class) and a component type. SimProvider class creates two methods that can be used to obtain the component class and type of any SimComponent as shown above.

3.3 Simulation Component

3.3.1 Components

The component is the basic building block of the simulator. There are different classes of components, e.g. StaticLink, GGSN, SGSN and MobileBTE. Some classes allow different types within the class in order to put up the simulation of a variety of implementations. Every component consists of an action routine and a data structure. All components of the same type share the same action routine; this routine is called each event that happens to a component. Each instance of a component has its own data structure, which is used to store up information that characterizes the component plus some standard information required by the simulator for every component.

3.4 Simulator Overview

The simulator can simulate whatever thing that can be modeled by a network of components that send messages to one another. The components schedule events for one another to cause things to happen. The model being simulated and the action of the components are entirely determined by the code controlling the components, not

3.4.2 Non-Functional Requirement

Following are some of the non-functional requirement of the simulator:

- Usability
    - The system should be user friendly. It will enhance and support rather than limit or restrict the understanding of routing functionality in UMTS core network. Human interfaces need to be intuitive and consistent with the user knowledge in order to let them gain some knowledge through the simulator.
- Flexibility
    - The system should have the capabilities to take advantage of new technologies and resources. The system should be able to implement in a changing environment.
- Reliability
    - Reliability is the extent to which a system can be expected to perform its intended function with required precision and accuracy. Thus, the system should be reliable in performing its simulation function and network operation. For example, whenever a button is clicked, the system should be able to perform some functionality or generate some message or animation to inform the user what is happening.
- Manageability
    - The modules within the system should be easy to manage. This will make the maintenance and enhancement works simpler and not time consuming.

3.5 Summary

This chapter covers the major analysis on programming language and tools as well as the key features of the simulator especially on JaNetSim Simulator. Each of the UMTS core network modules has different roles in the routing environment. Lastly, this chapter concludes by presenting the functional requirements and non-functional requirements of the simulator model to be designed. Detail of the system design will be discussed in the following chapter.

Chapter 4 – System Design

This chapter describes the on the whole system architecture design, class design, algorithm design and simulator design overview. An overview of the simulation model is to focus on the object-oriented approach. Classes are built for each of the objects to have their personal methods and processes being implemented.

4.1 System Architecture Design

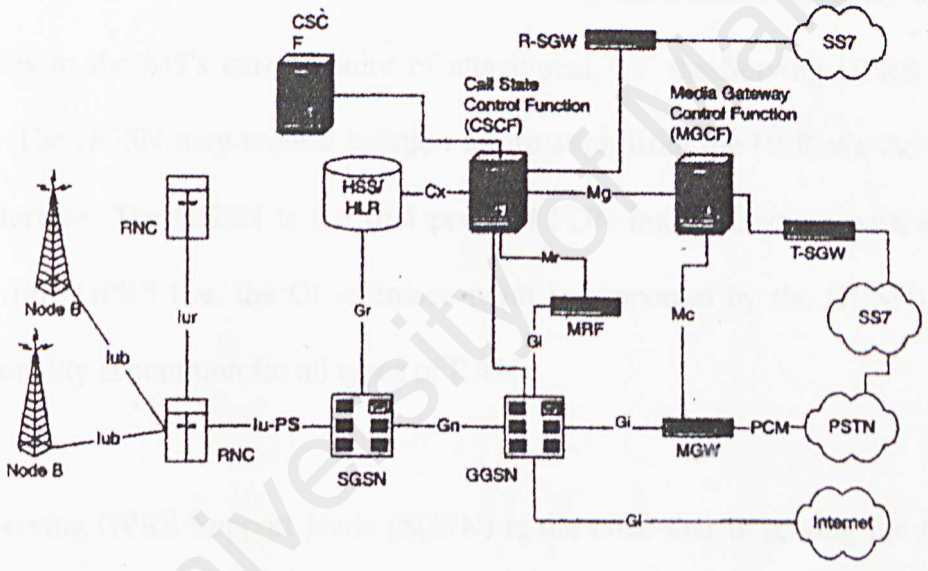


Figure 4-1: UMTS Core Network System Architecture

The packet domain uses packet-mode techniques to transfer high-speed and low-speed data and signalling in an efficient manner. The packet domain optimises the use of network and radio resources. Strict separation between the radio subsystem and network subsystem is maintained, allowing the network subsystem to be reused with other radio access technologies. [2]

The Serving GPRS Support Node (SGSN) keeps track of the location of an individual MS and performs security functions and access control. The SGSN is connected to the GERAN base station system through the Gb or Iu interface and/or to the UTRAN through the Iu interface. The SGSN also interfaces via the GPRS Service Switching Function with the GSM Service Control Function for optional CAMEL session and cost control service support.

The Gateway GPRS Support Node (GGSN) is the node that is accessed by the packet data network due to evaluation of the PDP address. It contains routeing information for PS-attached users. The routeing information is used to tunnel N-PDUs to the MS's current point of attachment, i.e. the Serving GPRS Support Node. The GGSN may request location information from the HLR via the optional Gc interface. The GGSN is the first point of PDN interconnection with a PLMN supporting GPRS (i.e. the Gi reference point is supported by the GGSN). GGSN functionality is common for all types of RANs.

The Serving GPRS Support Node (SGSN) is the node that is serving the MS. The SGSN supports GPRS for A/Gb mode (i.e. the Gb interface is supported by the SGSN) and/or Iu-mode (i.e. the Iu interface is supported by the SGSN). At PS attach, the SGSN establishes a mobility management context containing information pertaining to e.g. mobility and security for the MS. At PDP Context Activation, the SGSN establishes a PDP context, to be used for routeing purposes, with the GGSN that the subscriber will be using.

The Gateway GPRS Support Node (GGSN) provides interworking with packet data networks, and is connected with SGSNs via an IP-based packet domain PLMN backbone network.

GPRS shall support interworking with networks based on the Internet protocol (IP). IP is defined in RFC 791 [40]. The packet domain may provide compression of the TCP/IP header when an IP datagram is used within the context of a TCP connection. Mobile terminals offered service by a service provider may be globally addressable through the network operator's addressing scheme.

Registration is the means by which a user's Mobile Id is associated with the user's packet data protocol(s) and address (es) within the PLMN, and with the user's access point(s) to the packet data network. The association can be static, i.e. stored in an HLR, or dynamic, i.e. allocated on a per need basis.

Address translation is the conversion of one address to another address of a different type. Address translation may be used to convert a packet data network protocol address into an internal network address that can be used for routeing packets within and between the PLMN(s).

Address mapping is used to map a network address to another network address of the same type for the routeing and relaying of messages within and between the PLMN(s), for example to forward packets from one network node to another.A route is an ordered list of nodes used for the transfer of messages within and between the PLMN(s). Each route consists of the originating node, zero or more relay nodes and

the destination node. Routeing is the process of determining and using, in accordance with a set of rules, the route for transmission of a message within and between the PLMN(s).The mobility management functions are used to keep track of the current location of an MS within the PLMN or within another PLMN.

The IP multimedia core network (IM CN) subsystem enables PLMN operators to offer their subscribers multimedia services based on and built upon Internet applications, services and protocols. There is no intention here to standardise such services within the IM CN subsystem, the intention is that such services will be developed by PLMN operators and other third party suppliers including those in the Internet space using the mechanisms provided by the Internet and the IM CN subsystem. The IM CN subsystem should enable the convergence of, and access to, voice, video, messaging, data and web-based technologies for the wireless user, and combine the growth of the Internet with the growth in mobile communications.

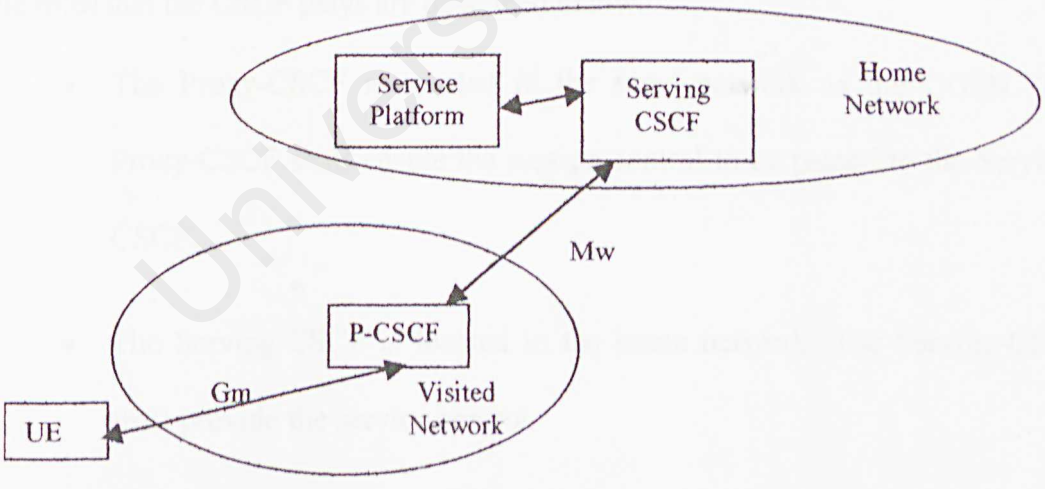


Figure 4-2: Service Platform in Home Network

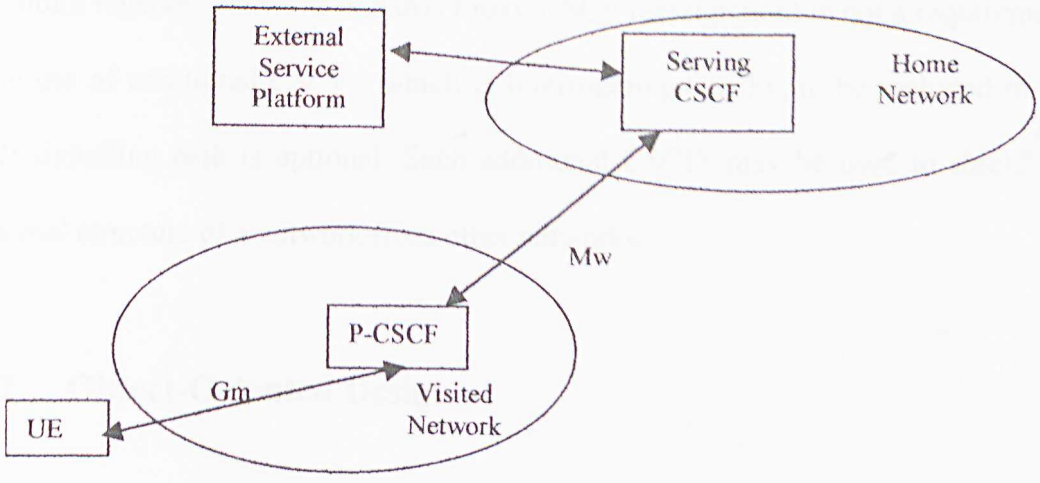


Figure 4-3: External Service Platform

There are two possible scenarios to provide services: [3]

- via the service platform in the Home Network
- via an external service platform (e.g. third party or visited network)

The box representing the external service platform could be located in either the visited network or in the 3rd party platform.

The roles that the CSCF plays are described below.

- The Proxy-CSCF is located in the same network as the GGSN. The Proxy-CSCF shall enable the session control to be passed to the Serving-CSCF.
- The Serving-CSCF is located in the home network. The Serving-CSCF shall provide the service control.

A Proxy-CSCF shall be supported in both roaming and non-roaming case, even when the Serving-CSCF is located in the same IM CN SS. Reassigning the Proxy-CSCF assigned during CSCF discovery is not a requirement in this release. Procedures to allow registration time Proxy-CSCF reassignment may be considered

in future releases. Network initiated Proxy-CSCF reassignment is not a requirement. The use of additional CSCFs, which is Interrogating-CSCFs, to be included in the SIP signalling path is optional. Such additional CSCFs may be used to shield the internal structure of a network from other networks.

4.2 Object-Oriented Design

UMTS Core Network will inherit all the features from SimComponent by creating WirelessLink, StaticLink, MobileBTE, StaticBTE, GGSN, SGSN and CSCF.

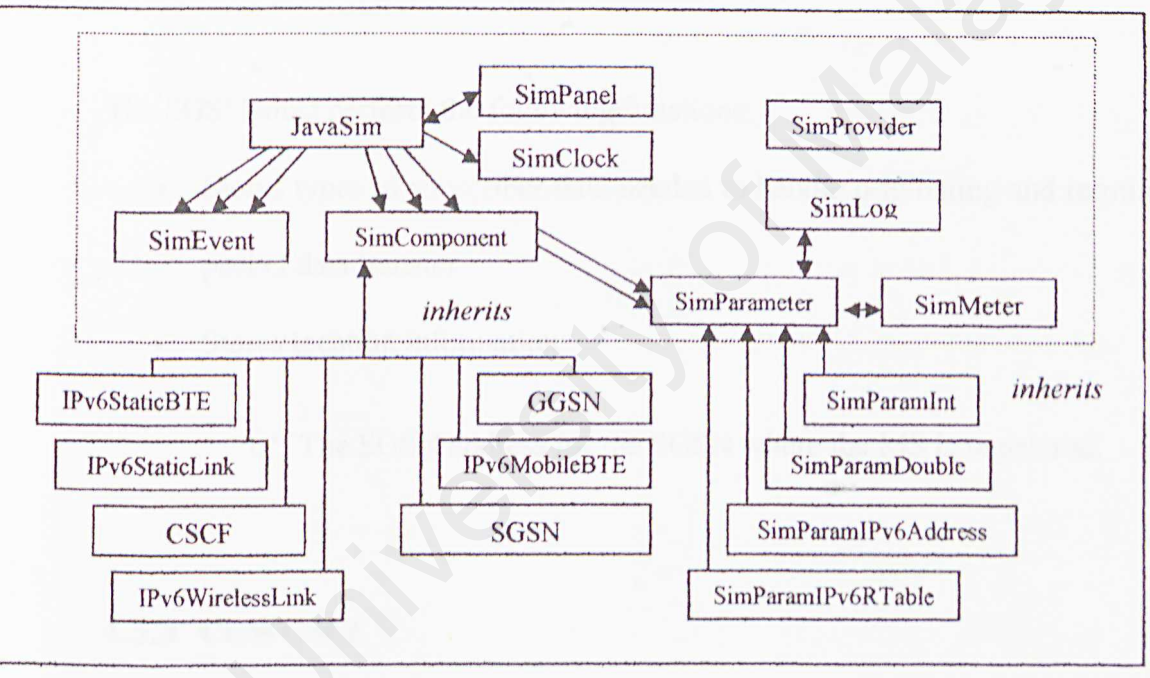


Figure 4-4: UMTS Core Network Simulator Objects

4.3 Class Design

This section gives a description on classes design in this project. Among the classes highlighted are *GGSN*, *SGSN* and *CSCF*.

4.3.1 Class GGSN

The GGSN must perform the following functions:

- Stores types of subscriber data needed to handle originating and terminating packet data transfer.
- Stores location information:
    - Depending on the routeing area where the UE is registered.

4.3.2 Class SGSN

The SGSN must perform the following functions:

- Stores types of subscriber data needed to handle originating and terminating packet data transfer.
- Stores location information:
    - The SGSN address for the SGSN where the MS is registered.

4.3.3 Class CSCF

The functions performed by the **P-CSCF** are:

- Forward the SIP register request received from the UE to an I-CSCF determined using the home domain name, as provided by the UE.

4.4 Program Flow Design

Three important algorithms are discussed here. First algorithm describes "Registration information flow – user not registered (CSCF)", second algorithm describes "Re-Registration information flow – user currently registered (CSCF)" [4] and last algorithm describes "Inter SGSN Routing Area Update" [2].

4.4.1 Registration information flow – user not registered (CSCF)

The application level registration can be initiated after the registration to the access is performed, and after IP connectivity for the signaling has been gained from the access network. For the purpose of the registration information flows, the subscriber is considered to be always roaming. For subscribers roaming in their home network, the home network shall perform the role of the visited network elements and the home network elements.

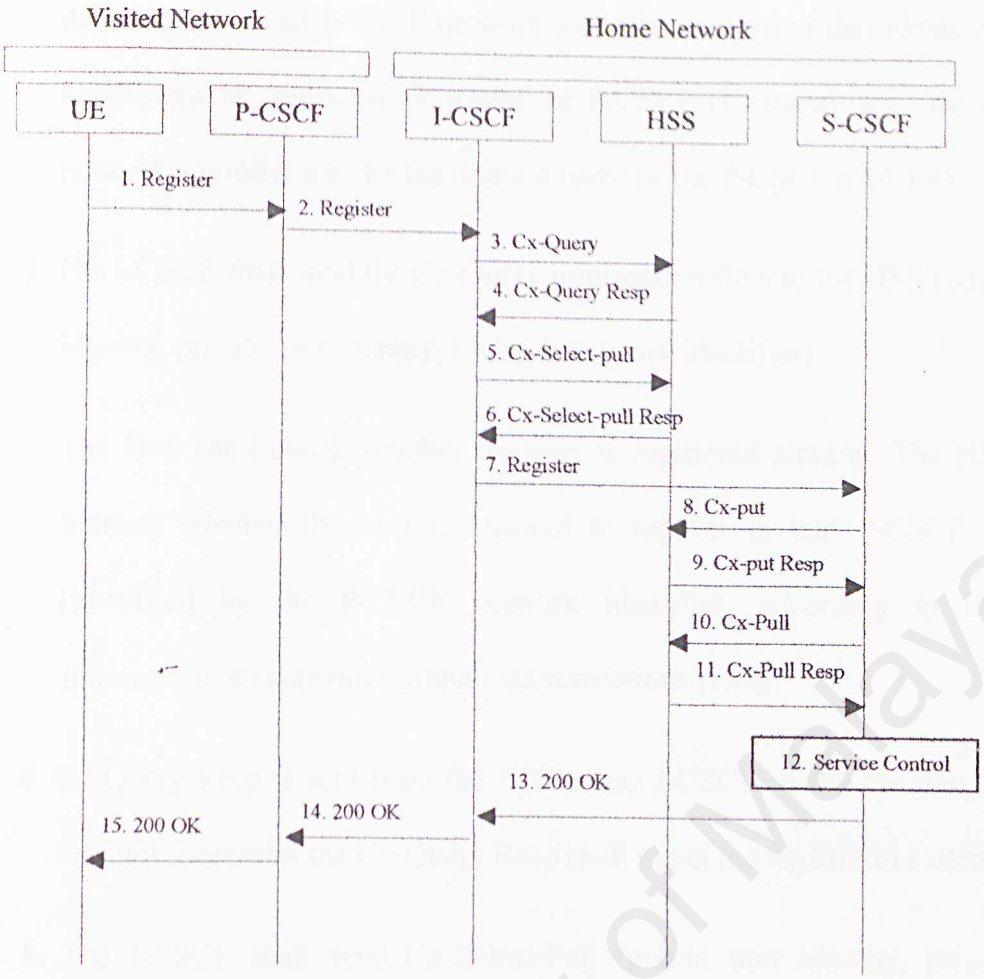


Figure 4-5: Registration – User not registered

1. After the UE has obtained a signalling channel through the access network, it can perform the IM registration. To do so, the UE sends the Register information flow to the proxy (public user identity, private user identity, home network domain name, UE IP address).

2. Upon receipt of the register information flow, the P-CSCF shall examine the "home domain name" to discover the entry point to the home network (i.e. the I-CSCF). The proxy shall send the Register information flow to the I-CSCF (P-CSCF address/name, public user identity, private user identity, P-CSCF network identifier, UE IP address). A name-address resolution mechanism is utilised in order to determine the address of the home network from the home

domain name. The P-CSCF network identifier is a string that identifies at the home network, the network where the P-CSCF is located (e.g., the P-CSCF network identifier may be the domain name of the P-CSCF network).

3. The I-CSCF shall send the Cx-Query information flow to the HSS (public user identity, private user identity, P-CSCF network identifier).

   The HSS shall check whether the user is registered already. The HSS shall indicate whether the user is allowed to register in that P-CSCF network (identified by the P-CSCF network identifier) according to the User subscription and operator limitations/restrictions if any.

4. Cx-Query Resp is sent from the HSS to the I-CSCF. If the checking in HSS was not successful the Cx-Query Resp shall reject the registration attempt.

5. The I-CSCF shall send Cx-Select-Pull (public user identity, private user identity) to the HSS to request the information related to the required S-CSCF capabilities which shall be input into the S-CSCF selection function.

6. The HSS shall send Cx-Select-Pull Resp (required S-CSCF capabilities) to the I-CSCF.

7. The I-CSCF, using the name of the S-CSCF, shall determine the address of the S-CSCF through a name-address resolution mechanism. The I-CSCF also determines the name of a suitable home network contact point, possibly based on information received from the HSS. The home network contact point may either be the S-CSCF itself, or a suitable I-CSCF(THIG) in case network configuration hiding is desired. If an I-CSCF(THIG) is chosen as the home network contact point for implementing network configuration hiding, it may

be distinct from the I-CSCF that appears in this registration flow, and it shall be capable of deriving the S-CSCF name from the home contact information. I-CSCF shall then send the register information flow (P-CSCF address/name, public user identity, private user identity, P-CSCF network identifier, UE IP address, I-CSCF(THIG) in case network configuration hiding is desired) to the selected S-CSCF. The home network contact point will be used by the P-CSCF to forward session initiation signalling to the home network.

8. The S-CSCF shall send Cx-Put (public user identity, private user identity, S-CSCF name) to the HSS. The HSS stores the S-CSCF name for that subscriber.

9. The HSS shall send Cx-Put Resp to the I-CSCF to acknowledge the sending of Cx-Put.

10. On receipt of the Cx-Put Resp information flow, the S-CSCF shall send the Cx-Pull information flow (public user identity, private user identity) to the HSS in order to be able to download the relevant information from the subscriber profile to the S-CSCF. The S-CSCF shall store the P-CSCF address/name, as supplied by the visited network. This represents the address/name that the home network forwards the subsequent terminating session signalling to for the UE.

11. The HSS shall return the information flow Cx-Pull Resp (user information) to the S-CSCF. The user information passed from the HSS to the S-CSCF shall include one or more names/addresses information which can be used to access the platform(s) used for service control while the user is registered at this S-CSCF. The S-CSCF shall store the information for the indicated user. In

addition to the names/addresses information, security information may also be sent for use within the S-CSCF.

12. Based on the filter criteria, the S-CSCF shall send register information to the service control platform and perform whatever service control procedures are appropriate.

13. The S-CSCF shall return the 200 OK information flow (home network contact information) to the I-CSCF. If an I-CSCF is chosen as the home network contact point for implementing network configuration hiding, the I-CSCF shall encrypt the S-CSCF address in the home network contact information.

14. The I-CSCF shall send information flow 200 OK (home network contact information) to the P-CSCF. The I-CSCF shall release all registration information after sending information flow 200 OK.

15. The P-CSCF shall store the home network contact information, and shall send information flow 200 OK to the UE.

4.4.2 Re-Registration information flow – user currently registered (CSCF)

Periodic application level re-registration is initiated by the UE either to refresh an existing registration or in response to a change in the registration status of the UE. Re-registration follows the same process as defined in subclause 5.2.2.3 "Registration Information Flow – User not registered". When initiated by the UE,

based on the registration time established during the previous registration, the UE shall keep a timer shorter than the registration related timer in the network.

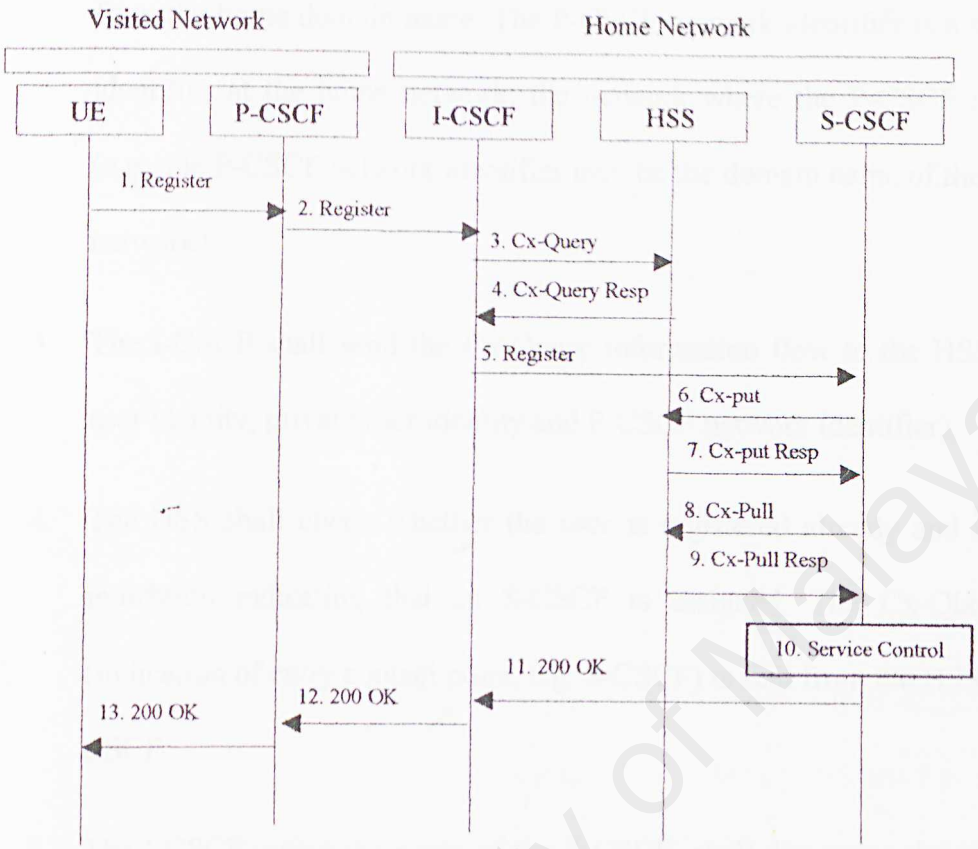


Figure 4-6: Re-registration - user currently registered

1. Prior to expiry of the agreed registration timer, the UE initiates a re-registration. To re-register, the UE sends a new REGISTER request. The UE sends the REGISTER information flow to the proxy (public user identity, private user identity, home network domain name, UE IP address).

2. Upon receipt of the register information flow, the P-CSCF shall examine the "home domain name" to discover the entry point to the home network (i.e. the I-CSCF). The proxy does not use the entry point cached from prior registrations. The proxy shall send the Register information flow to the I-CSCF (P-CSCF address/name, public user identity, private user identity, P-

CSCF network identifier, UE IP address). A name-address resolution mechanism is utilised in order to determine the address of the home network from the home domain name. The P-CSCF network identifier is a string that identifies at the home network, the network where the P-CSCF is located (e.g., the P-CSCF network identifier may be the domain name of the P-CSCF network).

3. The I-CSCF shall send the Cx-Query information flow to the HSS (public user identity, private user identity and P-CSCF network identifier).

4. The HSS shall check whether the user is registered already and return an indication indicating that an S-CSCF is assigned. The Cx-Query Resp (indication of entry contact point, e.g. S-CSCF) is sent from the HSS to the I-CSCF.

5. The I-CSCF, using the name of the S-CSCF, shall determine the address of the S-CSCF through a name-address resolution mechanism. The I-CSCF also determines the name of a suitable home network contact point, possibly based on information received from the HSS. The home network contact point may either be the S-CSCF itself, or a suitable I-CSCF(THIG) in case network configuration hiding is desired. If an I-CSCF(THIG) is chosen as the home network contact point for implementing network configuration hiding, it may be distinct from the I-CSCF that appears in this registration flow, and it shall be capable of deriving the S-CSCF name from the home contact information. I-CSCF shall then send the register information flow (P-CSCF address/name, public user identity, private user identity, P-CSCF network identifier, UE IP address, I-CSCF(THIG) in case network

configuration hiding is desired) to the selected S-CSCF. The home network contact point will be used by the P-CSCF to forward session initiation signalling to the home network.

6. The S-CSCF shall send Cx-Put (public user identity, private user identity, S-CSCF name) to the HSS. The HSS stores the S-CSCF name for that subscriber. Note: Optionally as an optimisation, the S-CSCF can detect that this is a re-registration and omit the Cx-Put request.

7. The HSS shall send Cx-Put Resp to the S-CSCF to acknowledge the sending of Cx-Put.

8. On receipt of the Cx-Put Resp information flow, the S-CSCF shall send the Cx-Pull information flow (public user identity, private user identity) to the HSS in order to be able to download the relevant information from the subscriber profile to the S-CSCF. The S-CSCF shall store the P-CSCF address/name, as supplied by the visited network. This represents the address/name that the home network forwards the subsequent terminating session signalling to for the UE. Note: Optionally as an optimisation, the S-CSCF can detect that this a re-registration and omit the Cx-Pull request.

9. The HSS shall return the information flow Cx-Pull-Resp (user information) to the S-CSCF. The S-CSCF shall store the user information for that indicated user.

10. Based on the filter criteria, the S-CSCF shall send re-registration information to the service control platform and perform whatever service control procedures are appropriate.

11. The S-CSCF shall return the 200 OK information flow (home network contact information) to the I-CSCF. If an I-CSCF is chosen as the home network contact point for implementing network configuration hiding, the I-CSCF shall encrypt the S-CSCF address in the home network contact information.

12. The I-CSCF shall send information flow 200 OK (home network contact information) to the P-CSCF. The I-CSCF shall release all registration information after sending information flow 200 OK.

13. The P-CSCF shall store the home network contact information, and shall send information flow 200 OK to the UE.

4.4.3 Inter SGSN Routing Area Update

The Inter SGSN Routing Area Update procedure is illustrated in below.

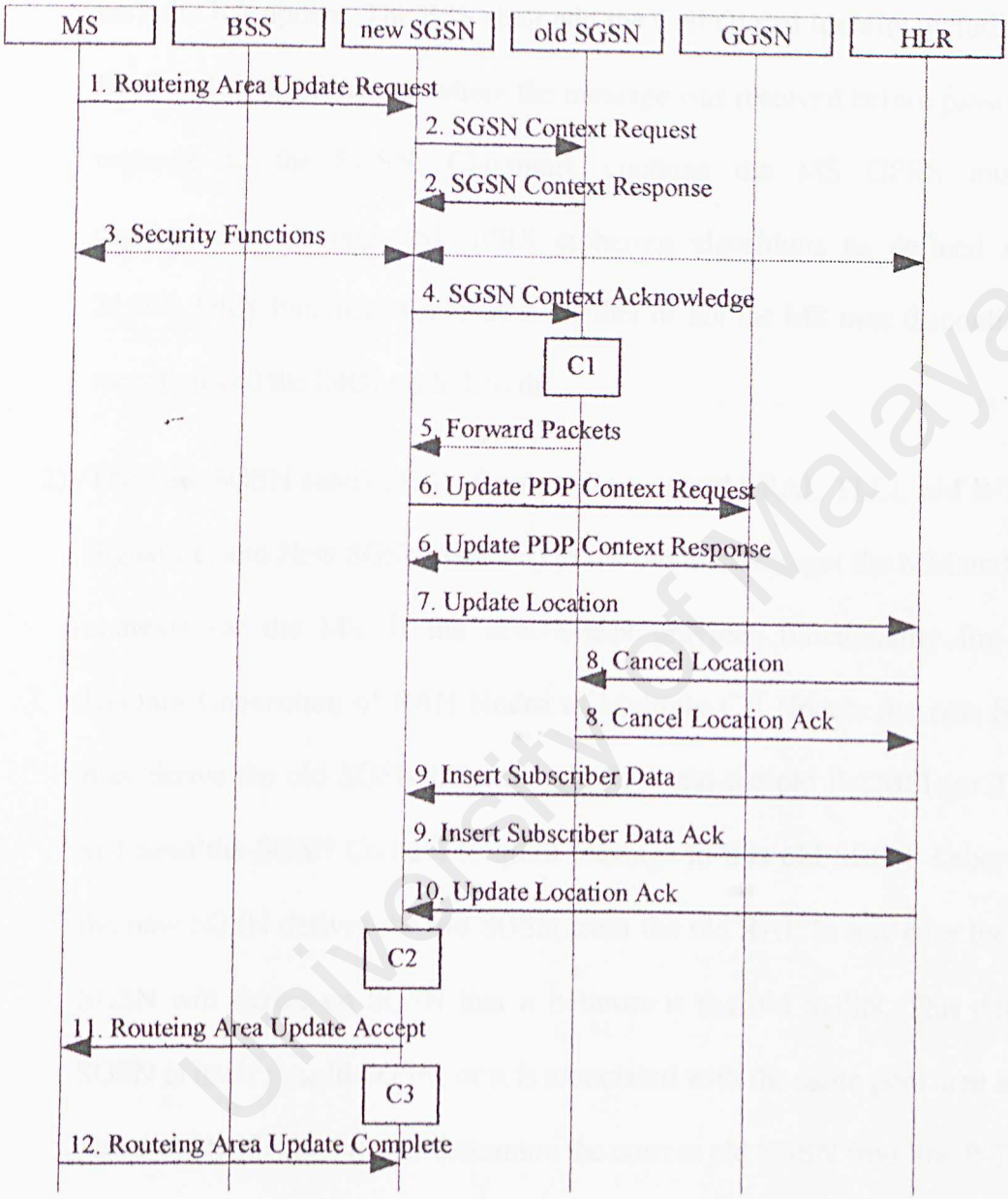


Figure 4-7: Inter SGSN Routing Area Update Procedure

1) The MS sends a Routeing Area Update Request (old RAI, old P-TMSI Signature, Update Type, Classmark, DRX parameters and MS Network Capability) to the new SGSN. Update Type shall indicate RA update or periodic RA update. The BSS shall add the Cell Global Identity including the RAC and LAC of the cell where the message was received before passing the message to the SGSN. Classmark contains the MS GPRS multislot capabilities and supported GPRS ciphering algorithms as defined in TS 24.008. DRX Parameters indicates whether or not the MS uses discontinuous reception and the DRX cycle length.

2) The new SGSN sends SGSN Context Request (old RAI, TLLI, old P-TMSI Signature, and New SGSN Address) to the old SGSN to get the MM and PDP contexts for the MS. If the new SGSN provides functionality for Intra Domain Connection of RAN Nodes to Multiple CN Nodes, the new SGSN may derive the old SGSN from the old RAI and the old P-TMSI (or TLLI) and send the SGSN Context Request message to this old SGSN. Otherwise, the new SGSN derives the old SGSN from the old RAI. In any case the new SGSN will derive an SGSN that it believes is the old SGSN. This derived SGSN is itself the old SGSN, or it is associated with the same pool area as the actual old SGSN and it will determine the correct old SGSN from the P-TMSI (or TLLI) and relay the message to that actual old SGSN. The old SGSN validates the old P-TMSI Signature and responds with an appropriate error cause if it does not match the value stored in the old SGSN.

3) Security functions may be executed. These procedures are defined in clause "Security Function". Ciphering mode shall be set if ciphering is supported.

If the security functions fail (e.g. because the SGSN cannot determine the HLR address to establish the Send Authentication Info dialogue), the Inter SGSN RAU Update procedure fails. A reject shall be returned to the MS with an appropriate cause.

4) The new SGSN sends an SGSN Context Acknowledge message to the old SGSN. This informs the old SGSN that the new SGSN is ready to receive data packets belonging to the activated PDP contexts. The old SGSN marks in its context that the MSC/VLR association and the information in the GGSNs and the HLR are invalid. This triggers the MSC/VLR, the GGSNs, and the HLR to be updated if the MS initiates a routeing area update procedure back to the old SGSN before completing the ongoing routeing area update procedure. If the security functions do not authenticate the MS correctly, then the routeing area update shall be rejected, and the new SGSN shall send a reject indication to the old SGSN. The old SGSN shall continue as if the SGSN Context Request was never received.

5) The old SGSN duplicates the buffered N-PDUs and starts tunnelling them to the new SGSN. Additional N-PDUs received from the GGSN before the timer described in step 2 expires are also duplicated and tunnelled to the new SGSN. N-PDUs that were already sent to the MS in acknowledged mode and that are not yet acknowledged by the MS are tunnelled together with the SNDCP N-PDU number. No N-PDUs shall be forwarded to the new SGSN after expiry of the timer described in step 2.

6) The new SGSN sends Update PDP Context Request (new SGSN Address, TEID, QoS Negotiated) to the GGSNs concerned. The GGSNs update their PDP context fields and return Update PDP Context Response (TEID).

7) The new SGSN informs the HLR of the change of SGSN by sending Update Location (SGSN Number, SGSN Address, and IMSI) to the HLR.

8) The HLR sends Cancel Location (IMSI, Cancellation Type) to the old SGSN with Cancellation Type set to Update Procedure. If the timer described in step 2 is not running, the old SGSN removes the MM and PDP contexts. Otherwise, the contexts are removed only when the timer expires. This allows the old SGSN to complete the forwarding of N-PDUs. It also ensures that the MM and PDP contexts are kept in the old SGSN in case the MS initiates another inter-SGSN routeing area update before completing the ongoing routeing area update to the new SGSN. The old SGSN acknowledges with Cancel Location Ack (IMSI).

9) The HLR sends Insert Subscriber Data (IMSI, GPRS Subscription Data) to the new SGSN. The new SGSN validates the MS's presence in the (new) RA. If due to regional subscription restrictions the MS is not allowed to be attached in the RA, the SGSN rejects the Routeing Area Update Request with an appropriate cause, and may return an Insert Subscriber Data Ack (IMSI, SGSN Area Restricted) message to the HLR. If all checks are successful, the SGSN constructs an MM context for the MS and returns an Insert Subscriber Data Ack (IMSI) message to the HLR.

10) The HLR acknowledges the Update Location by sending Update Location Ack (IMSI) to the new SGSN.

11) The new SGSN validates the MS's presence in the new RA. If due to roaming restrictions the MS is not allowed to be attached in the SGSN, or if subscription checking fails, the new SGSN rejects the routeing area update with an appropriate cause. If all checks are successful, the new SGSN constructs MM and PDP contexts for the MS. A logical link is established between the new SGSN and the MS. The new SGSN responds to the MS with Routeing Area Update Accept (P-TMSI, P-TMSI Signature, and Receive N-PDU Number). Receive N-PDU Number contains the acknowledgements for each acknowledged-mode NSAPI used by the MS, thereby confirming all mobile-originated N-PDUs successfully transferred before the start of the update procedure.

12) The MS acknowledges the new P-TMSI by returning a Routeing Area Update Complete (Receive N-PDU Number) message to the SGSN. Receive N-PDU Number contains the acknowledgements for each acknowledged-mode NSAPI used by the MS, thereby confirming all mobile-terminated N-PDUs successfully transferred before the start of the update procedure. If Receive N-PDU Number confirms reception of N-PDUs that were forwarded from the old SGSN, these N-PDUs shall be discarded by the new SGSN. LLC and SNDCP in the MS are reset.

4.5 Simulator Design Overview

Simulation model would firstly introduce the simulator architecture design that provides information for the user to produce network topologies using simulated ATM switches, terminal equipment and physical links.

Features that available in the proposed simulator include a mixture of applications, the behavior of which will determine the kind of traffic generated for transmission through the network. Users have the capability to control the parameters associated with these components, define the routes, and specify many details regarding the logging and display of performance data.

User interface to the simulator is through a JAVA application display screen simultaneously displays the network configuration, a control panel for running the simulation, and parameter information. The display contains a text window for user prompts, which also provides a place for parameter data entry. Output parameter values may be displayed in numerical form in "information windows" or as graphical "meters". Output parameter values may also be tagged for logging to a file; the data logging frequency is user dependent.

4.5.1 Graphical User Interface Design

Interface can advance the efficiency and effectiveness of the user when using the simulator. Thus, the interface design for the UMTS Core Network simulator must be simple to understand and easy to use. The users no need to keep in mind any dos

commands and what he or she needs to do is just some mouse click. We have to create the interface design as friendly as possible. The aim of this design is able to avoid failures and inappropriate procedures.

4.5.2 Design of Screen

The design of the graphical user interface display for UMTS Core Network simulator is divided into 3 main parts:

1. A network window to present UMTS Core Network configurations. This window is used both while creating the configurations and to explain network activity while the simulation is running.
2. A text window for messages that will prompt the user, and to provide a place for the user to key in text or parameter values.
3. A control panel that consists of a clock and several control buttons, such as START, RESET, RESUME and etc.

4.5.3 The Network Windows

The default setting for network windows is a blank area where the user can start built their own network topology in the area. To use the function of the system, the user just need to click on the selected tasks that are available such as button to create UMTS application, GGSN, network segment, B-TE, CSCF components and physical links.

The physical links are also considered components and are recognized, but they are represented on the figure by straight lines. The link between a B-TE and an UMTS Application is represented by a line but is not considered a component, i.e. it is not a physical entity and has no associated parameters.

When creating or modifying a component, an information window will appear beside its symbol, displaying the component's parameters. When a simulation is running, one or more meters may appear on the screen to display information about selected parameters.

4.5.4 The Text Windows

The text window appears as a bar at the bottom of the screen. The text window allows the program to present a variety of messages to the user. In addition, any keyboard input is displayed in the text window. The cursor does not need to be in the text window when entering information via the keyboard. When entering information using the keyboard, pressing, "Return" without entering any text will tell the program to recognize a default value or to abort that operation.

4.5.5 The Control Panel

The control panel appears on the right portion of the screen. It contains an analog clock, a digital clock and an array of control buttons. This button includes START, RESET, CONNECT MODE and PAUSE. The digital clock indicates the passage of

simulator time in a graphic style. The intent is not a precise timer but to offer the user an indication of how eventful the simulator is.

4.6 Expected Design Output

The design of output serves the intention of providing the information that the user needs, based on the criteria chosen by the users:

- User can build any type of network topology on the working space
- More than 1 B-TE can send request for address resolution where concurrent processing can execute each thread concurrently during the run time.
- The simulator can record on any information required by user to show the detail of the process during the simulation stage.

4.7 Summary

This chapter covers the major design issue for the UMTS Core Network simulator. This includes an overview of the system architecture which focuses on routing stages for the core network. The class design gives an illustration of the define attributes and functions for each class. In the following section, a description of the program flow design is presented. The major algorithm and the simulator flow are covered here.

Chapter 5 – Implementation

Chapter 5 discusses the implementation phases that need to be done for simulator. All the classes with important attributes will be revealed together with the explanation of these attributes as well as methods contained within the classes.

This UMTS Core Network simulator consists of main packages: JANETSIM. Following will describe attributes within classes of simulator UMTS Core Network components for this package. All this components are built on top of JANETSIM packages in the JANETSIM Simulator discussed in chapter 2 and chapter 3.

5.1 Simulator UMTS Core Network Component

Simulator UMTS Core Network Component consists of necessary classes for illustrating the UMTS Core Network architecture. Certain important classes which is useful for this purpose are *UMTS_GGSN.java, UMTS_CSCF.java, UMTS_Router.java, UMTS_TCP.java, UMTS_SwitchLink, UMTS_WirelessLink and UMTS_MobileBTE.*

UMTS_GGSN.java

```
package javasim;
import java.awt.*;
import java.io.Serializable;
```

```
class UMTS_GGSN extends IPv6Router implements Serializable {
  private MIPv6Processor myMIPv6Processor = null;
  private SimParamIPv6Address myHomeAgentAddress = null;
  private SimParamHomeAddressTable myHAtable=null;

//Constructor
  UMTS_GGSN(String name,int c,int t,JavaSim aSim,java.awt.Point loc) {
    super(name,c,t,aSim,loc);


    long ctick=theSim.now();
    myHomeAgentAddress = new SimParamIPv6Address("Gateway GPRS Support
Node IP Address",this, ctick, true);
    myHAtable=new SimParamHomeAddressTable("Gateway GPRS Support Node
Address Table",this,ctick, true);
    params.add(myHomeAgentAddress);
    params.add(myHAtable);

    isMobile = true;
  }
```

GGSN is the important component in UMTS Core Network simulator. GGSN act as router with further features to register user equipment information and location.

UMTS_CSCF.java

```
package javasim;
```

```
import java.awt.*;
import java.io.Serializable;

class UMTS_CSCF extends UMTS_Router implements Serializable {
  private MIPv6Processor myMIPv6Processor = null;
  private SimParamIPv6Address myHomeAgentAddress = null;
  private SimParamHomeAddressTable myHAtable=null;

//Constructor
  UMTS_CSCF(String name,int c,int t,JavaSim aSim,java.awt.Point loc) {
    super(name,c,t,aSim,loc);

    long ctick=theSim.now();
    myHomeAgentAddress = new SimParamIPv6Address("Call Session Control
Function IP Address",this, ctick, true);
    myHAtable=new SimParamHomeAddressTable("Call Session Control Function
Address Table",this,ctick, true);
    params.add(myHomeAgentAddress);
    params.add(myHAtable);

    isMobile = true;
  }
```

CSCF is the important component in UMTS Core Network simulator. CSCF act as router for voice purpose with further features to register user equipment information and location.

UMTS_Router.java

```
package javasim;
import java.awt.*;
import java.io.Serializable;

class UMTS_Router extends SimNetworkComp implements Serializable {
  private SimParamInterfaceTable myITable=null;
  //private MIPv6Processor myMIPv6Processor = null;

  protected ICMPv6Processor myICMPProcessor = null;
  protected RIPv6Processor myRIPv6Processor = null;
  protected SimParamIPv6RTable myRTable=null;
  protected SimParamInt myPacketHandled  = null;
  protected SimParamInt myMemoryUsage  = null;

 //Constructor
  UMTS_Router(String name,int c,int t,JavaSim aSim,java.awt.Point loc) {
    super(name,c,t,aSim,loc);

    long ctick=theSim.now();
    myRTable=new SimParamIPv6RTable("Routing Table",this,ctick);
```

```
    myITable=new SimParamInterfaceTable("Interfaces",this,ctick,true);
    myPacketHandled=new
SimParamInt("Packet Handled",getName(),ctick,true,false,0);
    myMemoryUsage=new
SimParamInt("Memory Usage (bytes)",getName(),ctick,true,false,0);

    params.add(myRTable);
    params.add(myITable);

    params.add(myPacketHandled);
    params.add(myMemoryUsage);

    isRouter = true;
    isMobile = false;
  }
```

Router is the important component in UMTS Core Network simulator. Routers route data and voice traffic to specific destination. It also exchange routing table with other routers by using RIP protocol.

UMTS_TCP.java

```
package javasim;
import java.awt.*;
import java.io.Serializable;
```

```
class UMTS_TCP extends SimComponent implements Serializable {
 class token implements Serializable
 {
  int seqNumber = 0;
  boolean ack = false;
 }

 private final int maxTokenQueueSize = 20;
 private int seqNumberNow = -1;
 private java.util.List tokenQueue = null;
 private SimComponent outBTE;
 private SimParamIPv6Address peerIP=null;
 private SimParamInt myTCPStartTime  = null;
 private SimParamInt myTCPSendInterval  = null;
 private SimParamInt myPacketMissed  = null;
 private SimParamDouble myPacketLatency  = null;

//Constructor
 UMTS_TCP(String name,int c,int t,JavaSim aSim,java.awt.Point loc) {
   super(name,c,t,aSim,loc);

   long ctick=theSim.now();
   peerIP=new SimParamIPv6Address("Search IP",this, ctick, true);
   myTCPStartTime=new
SimParamInt("Start Time (ms)",getName(),ctick,false,true,1000);
```

```
    myTCPSendInterval=new
SimParamInt("Send Interval (ms)",getName(),ctick,false,true,1000);
    myPacketMissed=new
SimParamInt("Packet Dropped",getName(),ctick,true,false,0);
    myPacketLatency=new
SimParamDouble("Round-trip Delay (us)",getName(),ctick,true,false,0);

    params.add(peerIP);
    params.add(myTCPStartTime);
    params.add(myTCPSendInterval);
    params.add(myPacketMissed);
    params.add(myPacketLatency);

    tokenQueue = new java.util.ArrayList();
  }
```

TCP is the important component in UMTS Core Network simulator. TCP is to control the fourth layer in OSI Model. It also search IP for another user in this simulator.

UMTS_SwitchLink.java

```
package javasim;
import java.awt.*;
import java.io.Serializable;
```

```
class UMTS_SwitchLink extends IPv6Link implements Serializable {
  private final int DEFAULT_SPEED = 10000; // unit: Kbps
  private final int DEFAULT_DISTANCE = 100; // unit: meter
  private final int DEFAULT_SWITCHING_DELAY = 10; // unit: us
  private final int DEFAULT_WIRED_CHANNEL = 1;

//Constructor
  UMTS_SwitchLink(String name,int c,int t,JavaSim aSim,java.awt.Point loc) {
    super(name,c,t,aSim,loc);

    mySpeed.setValue(DEFAULT_SPEED);
    myDistance.setValue(DEFAULT_DISTANCE);
    mySwitchingDelay.setValue(DEFAULT_SWITCHING_DELAY);
    mySupportedChannel.setValue(DEFAULT_WIRED_CHANNEL);
  }
```

Switch is the important component in UMTS Core Network simulator. Switch is to control the third layer in OSI Model. It also makes connection between other components like router, GGSN and CSCF in this simulator.

UMTS_WirelessLink.java

```
package javasim;
import java.awt.*;
import java.io.Serializable;
```

```
class UMTS_WirelessLink extends IPv6Link implements Serializable {
 private final int DEFAULT_SPEED = 10; // unit: Kbps
 private final int DEFAULT_DISTANCE = 100; // unit: meter
 private final int DEFAULT_SWITCHING_DELAY = 50; // unit: us
 private final int DEFAULT_WIRELESS_CHANNEL = 8;

 private SimParamInt myRange = null;
 private SimParamInt myBroadcastInterval = null;
 private SimParamInt myBTSID = null;

//Constructor
 UMTS_WirelessLink(String name,int c,int t,JavaSim aSim,java.awt.Point loc) {
  super(name,c,t,aSim,loc);

  long ctick=theSim.now();
  myRange = new SimParamInt("Coverage",getName(),ctick,false,true,100);
  myBroadcastInterval = new SimParamInt("Broadcast Beacon Interval
(ms)",getName(),ctick,false,true,DEFAULT_BROADCASTBEACONINTERVAL);
  myBTSID = new SimParamInt("Station ID",getName(),ctick,false,true,0);

  params.add(myRange);
  params.add(myBroadcastInterval);
  params.add(myBTSID);

  mySpeed.setValue(DEFAULT_SPEED);
```

```
    myDistance.setValue(DEFAULT_DISTANCE);
    mySwitchingDelay.setValue(DEFAULT_SWITCHING_DELAY);
    mySupportedChannel.setValue(DEFAULT_WIRELESS_CHANNEL);
  }
```

Base Station is the important component in UMTS Core Network simulator. Base Station is to control certain area in simulator for user equipment to communicate with others user equipment.

UMTS_MobileBTE.java

```
package javasim;
import java.awt.*;
import java.io.Serializable;

class UMTS_MobileBTE extends IPv6BTE implements Serializable {

private final int maxMissBeaconBroadcast = 10;
private final int movementInterval = 200; // unit: msec
private final double hysteriaThreshold = 1.10;
//private final int myIPCheckingInterval = 100; // unit: msec
private final int MAX_BU_RATE = 1; // unit: sec
private final int LIFETIME_CORRESPONDING_NODE = 5; // unit: sec
private final int LIFETIME_FOREIGN_AGENT = 5; // unit: sec
private final int DEFAULT_REPORT_INTERVAL = 500; // unit: msec
private final int MOBILEBTE_PACKET_COLOR = 0x808000;
```

```
private final int MAX_OUTQUEUE_SIZE = 10;

private final int DEFAULT_START_MOVING_DELAY = 40; // unit: sec


private SignalReportProcessor mySignalReportProcessor = null;

//private java.util.List myPreviousIPs = null;

//private java.util.List myPreviousDefaultGWs = null;

//private java.util.List myCorrespondingNodes = null;

private SimParamAddressExpiryTimeTable myPreviousDefaultGWs = null;

private SimParamAddressExpiryTimeTable myCorrespondingNodes = null;

private java.util.List myOutQueue = null;


private int missBeaconBroadcast = 0;

private int currentBroadcastBeaconInterval = 1000; // in MSec

private boolean switchingOver = false;

private boolean justSwitchOver = true;

private boolean waitingToSendBU = false;

private Point currentTarget = null;

private double currentSpeed = 0; // unit: pixel/sec

private int currentStep = 0;

private double xRemainder = 0;

private double yRemainder = 0;


private IPv6Address previousMSAddress = null;

private IPv6Address previousVCOA = null;

private long handoverStartTime = 0;
```

```
private SimParamFileChooser myScript = null;
private SimParamIPv6Address myHomeAgentAddress = null;
private SimParamIPv6Address myHomeAddress = null;
private SimParamInt myStartMovingDelay = null;
private SimParamInt mySerialNumber = null;
private SimParamDouble myHandoverLatency = null;

//Constructor
 UMTS_MobileBTE(String name,int c,int t,JavaSim aSim,java.awt.Point loc) {
  super(name,c,t,aSim,loc);

  long ctick=theSim.now();
  myStartMovingDelay=new SimParamInt("Start Moving Delay
(s)",getName(),ctick,false,true,DEFAULT_START_MOVING_DELAY);
  myScript = new SimParamFileChooser("Movement
Script",getName(),ctick,false,true);
  myHomeAgentAddress = new SimParamIPv6Address("Home Router IP
Address",this, ctick, true);
  myHomeAddress = new SimParamIPv6Address("User Equipment IP
Address",this, ctick, true);
  mySerialNumber=new SimParamInt("Mobile Serial
Number",getName(),ctick,false,false,0);
  myHandoverLatency=new SimParamDouble("Handover Latency
(us)",getName(),ctick,true,false,0);
```

```
    myPreviousDefaultGWs = new SimParamAddressExpiryTimeTable("Previous
Gateway",
     this,
     SimClock.Sec2Tick(LIFETIME_FOREIGN_AGENT),
     getTheSim().now());
   myCorrespondingNodes = new SimParamAddressExpiryTimeTable("Corres.
Node",
     this,
     SimClock.Sec2Tick(LIFETIME_CORRESPONDING_NODE),
     getTheSim().now());

  params.add(myStartMovingDelay);
  params.add(myScript);
  params.add(myHomeAgentAddress);
  params.add(myHomeAddress);
  params.add(mySerialNumber);
  params.add(myHandoverLatency);
  params.add(myPreviousDefaultGWs);
  params.add(myCorrespondingNodes);

  isMobile = true;
  mySerialNumber.setValue(getName().hashCode());
}
```

User Equipment is the important component in UMTS Core Network simulator. User Equipment is to communicate with others user equipment in UMTS Core Network simulator.

5.2 Summary

This chapter gives a suggestion on how the implementation processes on the UMTS Core Network simulator were carried out. Class implementation explains the attributes of each class. The class implementation also explains the technique in each class.

Chapter 6 – Testing

Simulator testing is done in three parts. Component testing will test the UMTS Core Network components features, module testing and systems testing.

6.1 Component Testing

Component testing is done in several classes like *UMTS_GGSN.java* and *UMTS_CSCF.java.*

UMTS_GGSN.java

Testing on GGSN is easy. A new object is instantiated and assigned with target GGSN IP address. At the end, the value is printed out.

1. Instantiate object and insert data into it.

```
class UMTS_GGSN extends UMTS_Router implements Serializable {
 private MIPv6Processor myMIPv6Processor = null;
 private SimParamIPv6Address myHomeAgentAddress = null;
 private SimParamHomeAddressTable myHAtable=null;
}
```

2. Input data into GGSN object.

myGGSNAddress = new SimParamIPv6Address("Gateway GPRS Support Node IP Address",this, ctick, true, "1:0:0:0:0:0:0:A5E");

3. Display the output.

System.out.prinln(getName() + "" + myGGSNAddress);

Output:

Gateway GPRS Support Node IP Address 1:0:0:0:0:0:0:A5E

UMTS_CSCF.java

CSCF is tested by instantiated and assigned with target CSCF IP address. At the end, the value is printed out.

4. Instantiate object and insert data into it.

```
class UMTS_CSCF extends UMTS_Router implements Serializable {
 private MIPv6Processor myMIPv6Processor = null;
 private SimParamIPv6Address myHomeAgentAddress = null;
 private SimParamHomeAddressTable myHAtable=null;
}
```

5. Input data into CSCF object.

myCSCFAddress = new SimParamIPv6Address("Call Session Control Function IP Address",this, ctick, true, "12:0:0:0:0:0:0:B4C");

6. Display the output.

System.out.prinln(getName() + "" + myCSCFAddress);

Output:

Call Session Control Function IP Address 12:0:0:0:0:0:0:B4C

6.2 Module Testing

The major purposes of the simulator module testing are checking the interaction between classes during the routing process. Routing testing is carried out with the purpose to make sure that routing for RIP is correctly and successfully reaching desired destination.

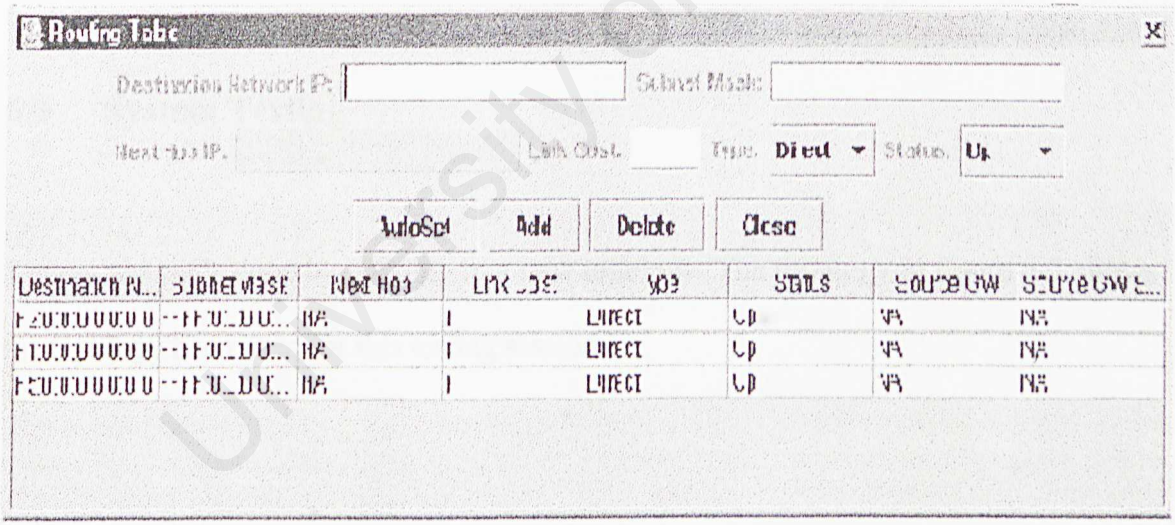


Figure 6-1: Routing Table before exchange RIP

Above is one of the routers before exchange routing table using RIP. The table has direct link only.

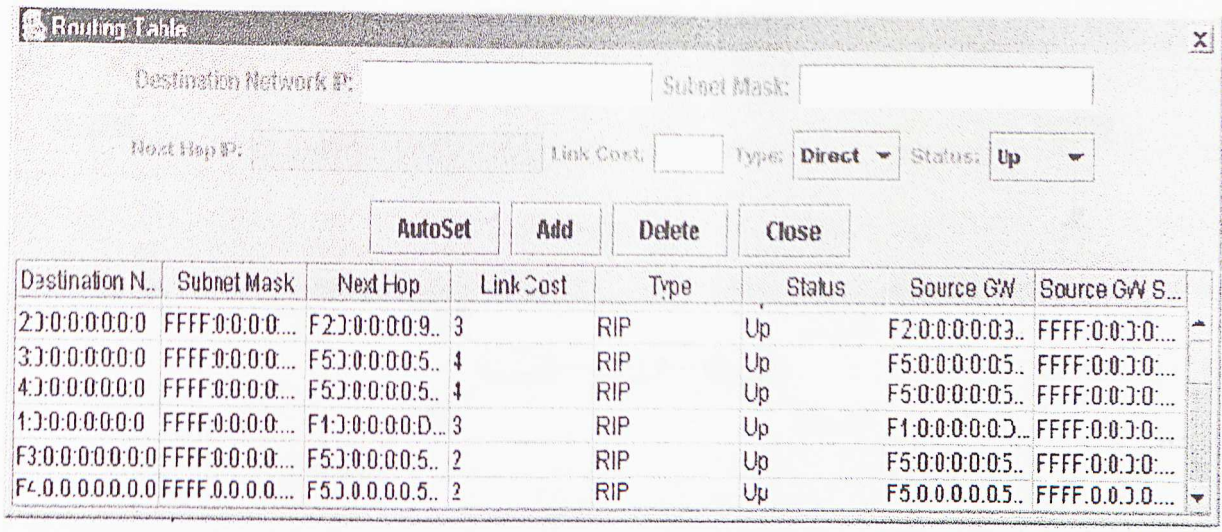


Figure 6-2: Routing Table after exchange RIP

Above is one of the routers after exchange routing table using RIP. The table has direct link and RIP type. Base on the routing table above, it shown that the routers in simulator successfully exchange their routing table with another.

6.3 System Testing

System testing is done by building in a new topology. The topology in figure 6-3 shows the topology been used for this testing session.

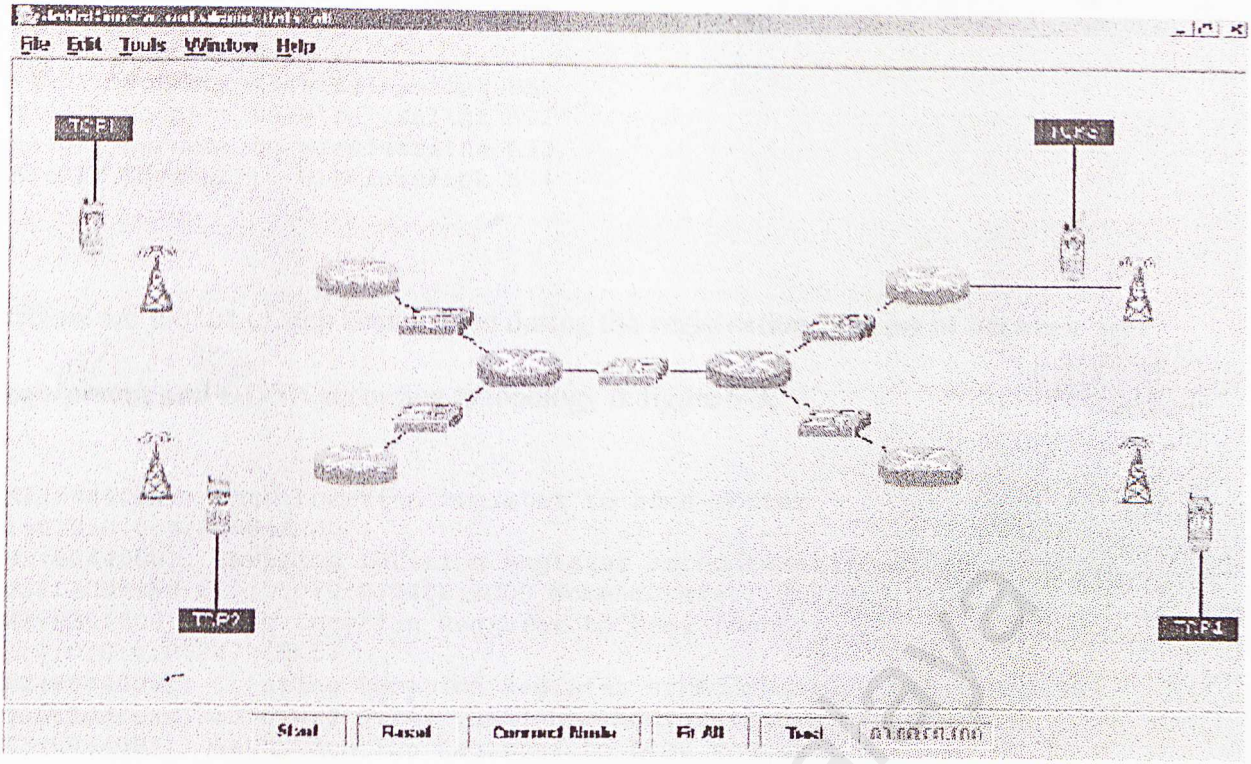


Figure 6-3: Testing Topology

Below are the list of step been logged during the **registration process simulation** according to topology in figure 6-3.

```
Hello, I am R1
Hello, I am R2
Hello, I am L1
Hello, I am L2
Hello, I am L3
Hello, I am L4
Hello, I am L5
Hello, I am GGSN1
Hello, I am GGSN2
Hello, I am GGSN3
Hello, I am GGSN4
Hello, I am M1
Hello, I am M2
Hello, I am M3
Hello, I am M4
Hello, I am BS1
Hello, I am BS2
Hello, I am BS3
Hello, I am BS4
Hello, I am TCP1
Hello, I am TCP2
```

```
Hello, I am TCP3
Hello, I am TCP4
M1(0): Connect to wirelesslink BS1
M2(0): Connect to wirelesslink BS2
M3(0): Connect to wirelesslink BS3
M4(0): Connect to wirelesslink BS4
```

Below are the list of step been logged during the **registration process of between user equipment and GGSN** according to topology in figure 6-3.

```
M1(8844000): Send Location Register to home router
1:0:0:0:0:0:206E:AC00
M1(8844000): Creating Location Register srcAddress:1:0:0:0:0:0:0:984
destAddress:1:0:0:0:0:0:206E:AC00 HomeAddress:1:0:0:0:0:0:0:984
M2(8844000): Send Location Register to home router
2:0:0:0:0:0:2479:BA57
M2(8844000): Creating Location Register srcAddress:2:0:0:0:0:0:0:985
destAddress:2:0:0:0:0:0:2479:BA57 HomeAddress:2:0:0:0:0:0:0:985
M3(8844000): Send Location Register to home router
3:0:0:0:0:0:2884:C8B0
M3(8844000): Creating Location Register srcAddress:3:0:0:0:0:0:0:986
destAddress:3:0:0:0:0:0:2884:C8B0 HomeAddress:3:0:0:0:0:0:0:986
M4(8844000): Send Location Register to home router
4:0:0:0:0:0:2C8F:D70B
M4(8844000): Creating Location Register srcAddress:4:0:0:0:0:0:0:987
destAddress:4:0:0:0:0:0:2C8F:D70B HomeAddress:4:0:0:0:0:0:0:987
BS1(8845000): Packet delay srcAddress:1:0:0:0:0:0:0:984
destAddress:1:0:0:0:0:0:206E:AC00 delay(us):41600
BS2(8845000): Packet delay srcAddress:2:0:0:0:0:0:0:985
destAddress:2:0:0:0:0:0:2479:BA57 delay(us):41600
BS3(8845000): Packet delay srcAddress:3:0:0:0:0:0:0:986
destAddress:3:0:0:0:0:0:2884:C8B0 delay(us):41600
BS4(8845000): Packet delay srcAddress:4:0:0:0:0:0:0:987
destAddress:4:0:0:0:0:0:2C8F:D70B delay(us):41600
GGSN1(13842000): Handling Location Register from COA:1:0:0:0:0:0:0:984
HomeAddress:1:0:0:0:0:0:0:984 expiry time:-1
GGSN1(13842000): Creating Binding Ack from:1:0:0:0:0:0:206E:AC00
to:1:0:0:0:0:0:0:984
GGSN2(13842000): Handling Location Register from COA:2:0:0:0:0:0:0:985
HomeAddress:2:0:0:0:0:0:0:985 expiry time:-1
GGSN2(13842000): Creating Binding Ack from:2:0:0:0:0:0:2479:BA57
to:2:0:0:0:0:0:0:985
GGSN3(13842000): Handling Location Register from COA:3:0:0:0:0:0:0:986
HomeAddress:3:0:0:0:0:0:0:986 expiry time:-1
GGSN3(13842000): Creating Binding Ack from:3:0:0:0:0:0:2884:C8B0
to:3:0:0:0:0:0:0:986
GGSN4(13842000): Handling Location Register from COA:4:0:0:0:0:0:0:987
HomeAddress:4:0:0:0:0:0:0:987 expiry time:-1
```

```
GGSN4(13842000): Creating Binding Ack from:4:0:0:0:0:0:2C8F:D70B
to:4:0:0:0:0:0:0:987
```

Below are the list of step been logged during the **registration process and exchange routing table between routers and GGSN** according to topology in figure 6-3.

```
R1(2366982657)->RIP: New Route
L2(2366983657): Packet delay srcAddress:F2:0:0:0:0:0:5F:1F5A
destAddress:FF02:0:0:0:0:0:0:9 delay(us):77
L1(2366983657): Packet delay srcAddress:F1:0:0:0:0:0:5F:153B
destAddress:FF02:0:0:0:0:0:0:9 delay(us):77
L5(2366983657): Packet delay srcAddress:F5:0:0:0:0:0:5F:3DB7
destAddress:FF02:0:0:0:0:0:0:9 delay(us):77
GGSN1(2366993657)->RIP: New Route
R2(2366993657)->RIP: New Route
L1(2366994657): Packet delay srcAddress:F1:0:0:0:0:0:D9A:1C12
destAddress:FF02:0:0:0:0:0:0:9 delay(us):77
L5(2366994657): Packet delay srcAddress:F5:0:0:0:0:0:5F:4720
destAddress:FF02:0:0:0:0:0:0:9 delay(us):77
L3(2366994657): Packet delay srcAddress:F3:0:0:0:0:0:5F:32E0
destAddress:FF02:0:0:0:0:0:0:9 delay(us):77
L4(2366994657): Packet delay srcAddress:F4:0:0:0:0:0:5F:3D00
destAddress:FF02:0:0:0:0:0:0:9 delay(us):77
GGSN3(2367004657)->RIP: New Route
GGSN4(2367004657)->RIP: New Route
```

Below are the list of step been logged during the **registration process and search IP between user equipment** according to topology in figure 6-3.

```
GGSN2(3308372595): srcIP:3:0:0:0:0:0:0:986 destIP:2:0:0:0:0:0:0:985
nextHop:2:0:0:0:0:0:0:985
BS2(3308373595): Packet delay srcAddress:3:0:0:0:0:0:0:986
destAddress:2:0:0:0:0:0:0:985 delay(us):836800
GGSN1(3338388994): srcIP:1:0:0:0:0:0:0:984 destIP:4:0:0:0:0:0:0:987
nextHop:F1:0:0:0:0:0:5F:153B
L1(3338389994): Packet delay srcAddress:1:0:0:0:0:0:0:984
destAddress:4:0:0:0:0:0:0:987 delay(us):837
R1(3338491194): srcIP:1:0:0:0:0:0:0:984 destIP:4:0:0:0:0:0:0:987
nextHop:F5:0:0:0:0:0:5F:4720
L5(3338492194): Packet delay srcAddress:1:0:0:0:0:0:0:984
destAddress:4:0:0:0:0:0:0:987 delay(us):837
R2(3338593394): srcIP:1:0:0:0:0:0:0:984 destIP:4:0:0:0:0:0:0:987
nextHop:F4:0:0:0:0:0:17B:DAD8
```

```
L4(3338594394): Packet delay srcAddress:1:0:0:0:0:0:0:984
destAddress:4:0:0:0:0:0:0:987 delay(us):837
GGSN4(3338695594): srcIP:1:0:0:0:0:0:0:984 destIP:4:0:0:0:0:0:0:987
nextHop:4:0:0:0:0:0:0:987
BS4(3338696594): Packet delay srcAddress:1:0:0:0:0:0:0:984
destAddress:4:0:0:0:0:0:0:987 delay(us):836800
L5(3343975944): Packet delay srcAddress:F5:0:0:0:0:0:5F:4720
destAddress:FF02:0:0:0:0:0:0:1 delay(us):37
TCP3(3344905580): TCP echo request:36
BS3(3344906580): Packet delay srcAddress:3:0:0:0:0:0:0:986
destAddress:2:0:0:0:0:0:0:985 delay(us):836800
L2(3348723208): Packet delay srcAddress:F2:0:0:0:0:0:990:656
destAddress:FF02:0:0:0:0:0:0:1 delay(us):37
L1(3350390237): Packet delay srcAddress:F1:0:0:0:0:0:5F:153B
destAddress:FF02:0:0:0:0:0:0:1 delay(us):37
L3(3386542643): Packet delay srcAddress:F3:0:0:0:0:0:585:F098
destAddress:FF02:0:0:0:0:0:0:1 delay(us):37
TCP1(3403846035): TCP echo request:39
BS1(3403847035): Packet delay srcAddress:1:0:0:0:0:0:0:984
destAddress:4:0:0:0:0:0:0:987 delay(us):
M2: Updating time in Corres. Node address:3:0:0:0:0:0:0:986
TCP2(3408795595): Got msg from TCP search IP:3:0:0:0:0:0:0:986
TCP2(3408795595): Responding to TCP echo request:35
BS2(3408796595): Packet delay srcAddress:2:0:0:0:0:0:0:985
destAddress:3:0:0:0:0:0:0:986 delay(us):44800
TCP1(3408880467): TCP echo request:40
```

6.4 Summary

This chapter gives an idea on how testing processes on the UMTS Core Network simulator were carried out. Testing of UMTS Core Network simulator begins with component testing and followed system testing. Component testing focuses on the individual testing for each class. Meanwhile, module testing focuses on routing testing. The system testing tests the whole simulator to ensure that it runs on the actual UMTS Core Network environment.

Chapter 7 – Conclusion

An object-oriented UMTS Core Network simulator environment is designed and implemented to fulfill the purpose of improving the effectiveness in performing computer network simulation studies. This simulation project is designed to be portable, which allows component developed by different modeler to be shared and thus accomplish reusability and flexibility. It is possible for different modeler to add or remove any methods currently designed since the simulator is built by an object-oriented approach using Java programming language. Development of additional component to extend the system for other purpose of simulation study could also be achieved but may require small modification to parts of the program codes in order to incorporate the new component for simulation needs.

A great benefit of Java is the multithreading technique, which is able to write programs with parallel activities. An UMTS Core Network simulator is highly dependent to a clock tick where executor classes could need to inter communicates in between each other. The period for a simulator to execute for each executor object is generally different. Therefore, one could need to wait for another object to complete their task before executing their next job. So, a clock tick is used to synchronize their jobs in between objects,

This project managed to achieve the overall project objectives and goals, i.e. development of an object-oriented, *m*ultithreading and provide a better graphical user interface for data capturing and better understanding for user to experience the simulator. Lastly, the following highlights strengths, limitations as well as the proposed future enhancements

System Strengths

- The simulator is fully object-oriented whereby all the functions and modules are built in class. In addition, problem of coupling is highly reduced.

- The simulator is built using Java Thread technology. When more than one switch is created, switching environment becomes more realistic where all switches run concurrently instead of sequentially.

- GGSN class, CSCF class and Router class are making used of message passing method in thorough out the entire simulation process. The syntax and logic are similar to Mobile IP application, therefore making the work easier.

System Limitation

- The simulator works based on Java Application, therefore this it is not support web-enable features.

- The simulator works based on RIP only. Other routing method is not presented.

- The simulator works based in limited user equipment in single base station and complex configuration in user equipment parameters.

- The Quality of Service (QoS) is not implemented to manage the bandwidth more efficient between data traffic and signaling traffic (for voice purposed) in UMTS Core Network environment.

Future Enhancements

- It should allow the execution of performance comparison for different types of routing method.

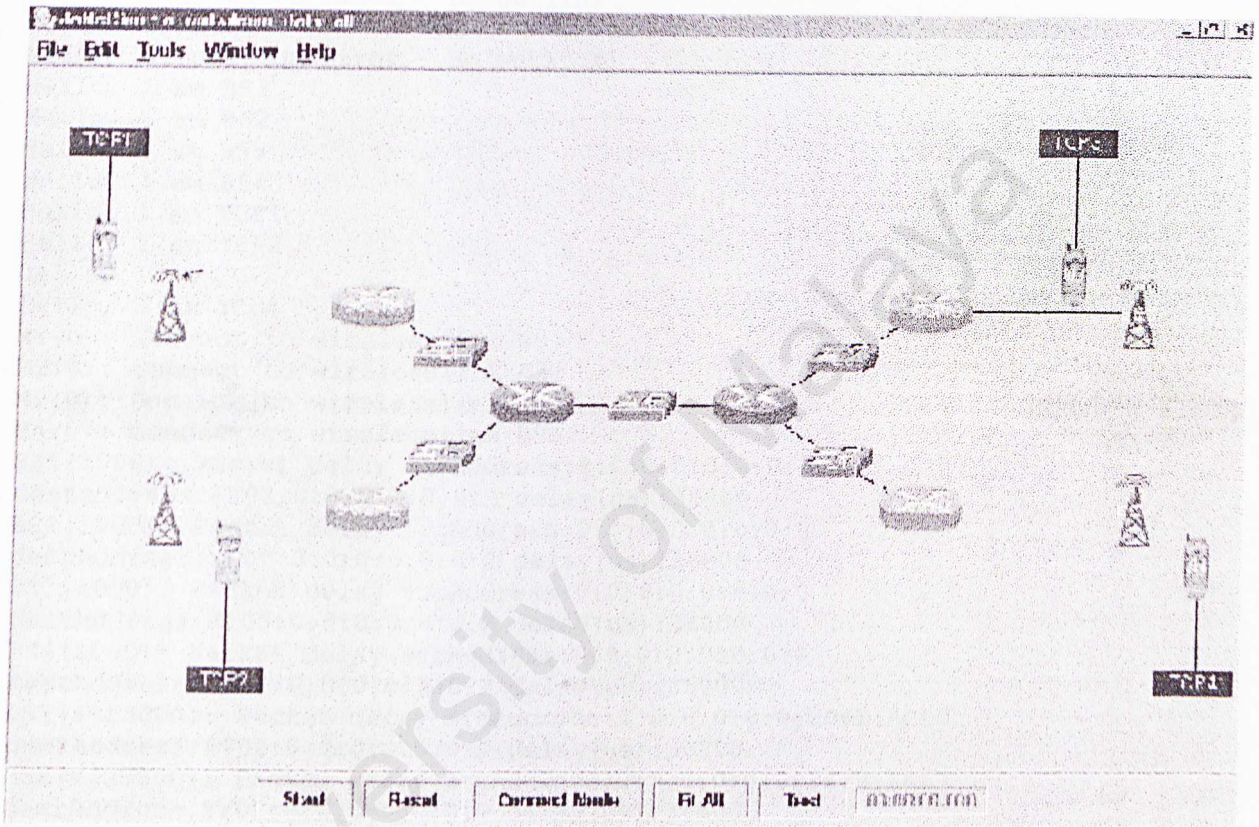
- It is hoped that the UMTS Core Network is extended to support more user equipment in single base station and have feature like autoset for user equipment parameter to make more easy for user to configure user equipment parameters.

- It should also include actual QoS feature in GGSN, CSCF and router.

From this project, the most valuable experience is to understand into more detail for the 3GPP UMTS Release 5 All-IP Network Architecture mainly for UMTS Core Network network. It is valuable to have experience at the same time building object-oriented model for the UMTS Core Network simulator, which mainly emphasizing on data and signaling traffic.

Appendix

Below is UMTS Core Network topology for **data traffic** which consist user equipment, base station, GGSN, switch, router and TCP application.



UMTS Core Network Topology for Data Traffic

Below are the list of step been logged during the **overall registration process simulation** according to UMTS Core Network topology for **data traffic**.

```
Hello, I am R1
Hello, I am R2
Hello, I am L1
Hello, I am L2
```

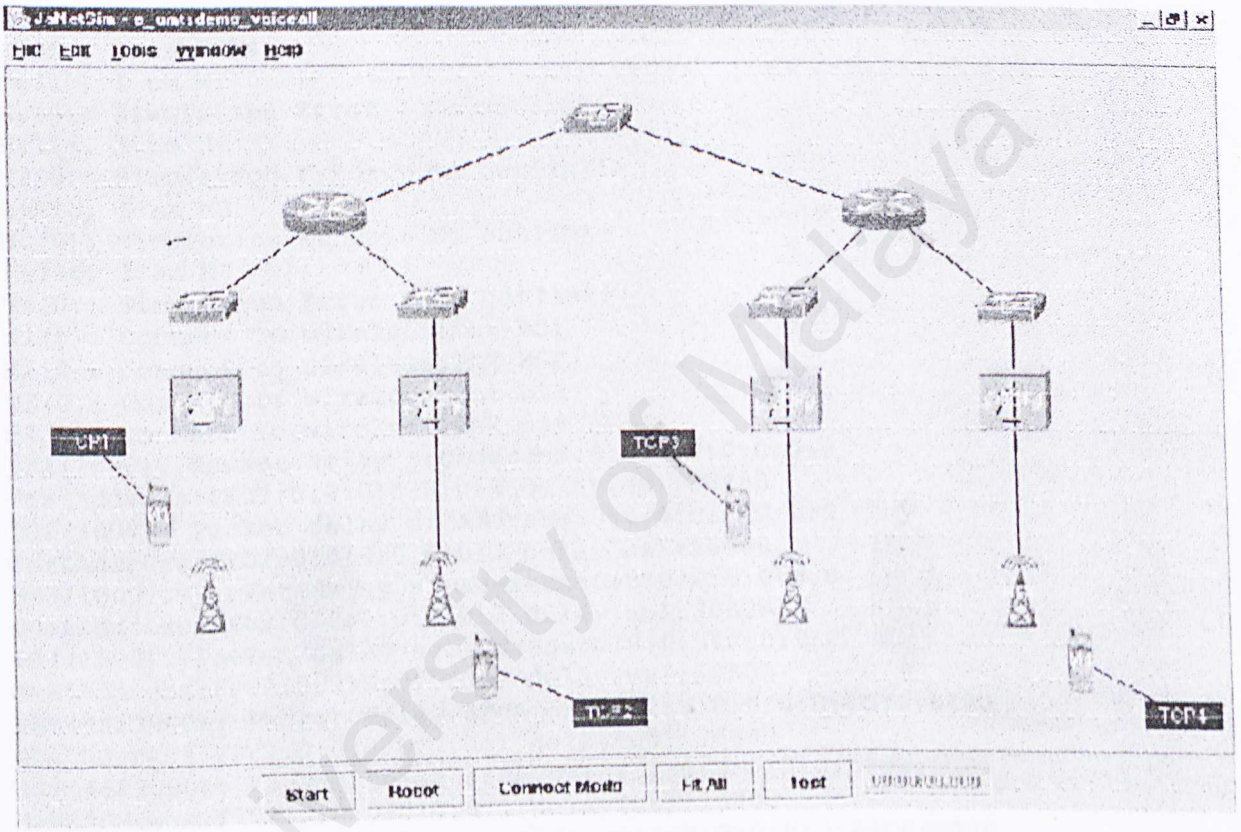
```
Hello, I am L3
Hello, I am L4
Hello, I am L5
Hello, I am GGSN1
Hello, I am GGSN2
Hello, I am GGSN3
Hello, I am GGSN4
Hello, I am M1
M1(0): Simulation Error - No outlink!
Hello, I am M2
M2(0): Simulation Error - No outlink!
Hello, I am M3
M3(0): Simulation Error - No outlink!
Hello, I am M4
M4(0): Simulation Error - No outlink!
Hello, I am BS1
Hello, I am BS2
Hello, I am BS3
Hello, I am BS4
Hello, I am TCP1
Hello, I am TCP2
Hello, I am TCP3
Hello, I am TCP4
M1(0): Connect to wirelesslink BS1
M2(0): Connect to wirelesslink BS2
M3(0): Connect to wirelesslink BS3
M4(0): Connect to wirelesslink BS4
BS1(1000): Packet delay srcAddress:0:0:0:0:0:0:0:0
destAddress:FF02:0:0:0:0:0:0:2 delay(us):36800
BS2(1000): Packet delay srcAddress:0:0:0:0:0:0:0:0
destAddress:FF02:0:0:0:0:0:0:2 delay(us):36800
BS3(1000): Packet delay srcAddress:0:0:0:0:0:0:0:0
destAddress:FF02:0:0:0:0:0:0:2 delay(us):36800
BS4(1000): Packet delay srcAddress:0:0:0:0:0:0:0:0
destAddress:FF02:0:0:0:0:0:0:2 delay(us):36800
BS1(4423000): Packet delay srcAddress:1:0:0:0:0:0:206E:AC00
destAddress:FF02:0:0:0:0:0:0:1 delay(us):36800
BS2(4423000): Packet delay srcAddress:2:0:0:0:0:0:2479:BA57
destAddress:FF02:0:0:0:0:0:0:1 delay(us):36800
BS3(4423000): Packet delay srcAddress:3:0:0:0:0:0:2884:C8B0
destAddress:FF02:0:0:0:0:0:0:1 delay(us):36800
BS4(4423000): Packet delay srcAddress:4:0:0:0:0:0:2C8F:D70B
destAddress:FF02:0:0:0:0:0:0:1 delay(us):36800
M1(8844000): Send Location Register to home router
1:0:0:0:0:0:206E:AC00
M1(8844000): Creating Location Register srcAddress:1:0:0:0:0:0:0:984
destAddress:1:0:0:0:0:0:206E:AC00 HomeAddress:1:0:0:0:0:0:0:984
M2(8844000): Send Location Register to home router
2:0:0:0:0:0:2479:BA57
M2(8844000): Creating Location Register srcAddress:2:0:0:0:0:0:0:985
destAddress:2:0:0:0:0:0:2479:BA57 HomeAddress:2:0:0:0:0:0:0:985
M3(8844000): Send Location Register to home router
3:0:0:0:0:0:2884:C8B0
M3(8844000): Creating Location Register srcAddress:3:0:0:0:0:0:0:986
destAddress:3:0:0:0:0:0:2884:C8B0 HomeAddress:3:0:0:0:0:0:0:986
```

```
M4(8844000): Send Location Register to home router
4:0:0:0:0:0:2C8F:D70B
M4(8844000): Creating Location Register srcAddress:4:0:0:0:0:0:0:987
destAddress:4:0:0:0:0:0:2C8F:D70B HomeAddress:4:0:0:0:0:0:0:987
BS1(8845000): Packet delay srcAddress:1:0:0:0:0:0:0:984
destAddress:1:0:0:0:0:0:206E:AC00 delay(us):41600
BS2(8845000): Packet delay srcAddress:2:0:0:0:0:0:0:985
destAddress:2:0:0:0:0:0:2479:BA57 delay(us):41600
BS3(8845000): Packet delay srcAddress:3:0:0:0:0:0:0:986
destAddress:3:0:0:0:0:0:2884:C8B0 delay(us):41600
BS4(8845000): Packet delay srcAddress:4:0:0:0:0:0:0:987
destAddress:4:0:0:0:0:0:2C8F:D70B delay(us):41600
GGSN1(13842000): Handling Location Register from COA:1:0:0:0:0:0:0:984
HomeAddress:1:0:0:0:0:0:0:984 expiry time:-1
GGSN1(13842000): Creating Binding Ack from:1:0:0:0:0:0:206E:AC00
to:1:0:0:0:0:0:0:984
GGSN2(13842000): Handling Location Register from COA:2:0:0:0:0:0:0:985
HomeAddress:2:0:0:0:0:0:0:985 expiry time:-1
GGSN2(13842000): Creating Binding Ack from:2:0:0:0:0:0:2479:BA57
to:2:0:0:0:0:0:0:985
GGSN3(13842000): Handling Location Register from COA:3:0:0:0:0:0:0:986
HomeAddress:3:0:0:0:0:0:0:986 expiry time:-1
GGSN3(13842000): Creating Binding Ack from:3:0:0:0:0:0:2884:C8B0
to:3:0:0:0:0:0:0:986
GGSN4(13842000): Handling Location Register from COA:4:0:0:0:0:0:0:987
HomeAddress:4:0:0:0:0:0:0:987 expiry time:-1
GGSN4(13842000): Creating Binding Ack from:4:0:0:0:0:0:2C8F:D70B
to:4:0:0:0:0:0:0:987
GGSN1(13842000): srcIP:1:0:0:0:0:0:206E:AC00 destIP:1:0:0:0:0:0:0:984
nextHop:1:0:0:0:0:0:0:984
GGSN2(13842000): srcIP:2:0:0:0:0:0:2479:BA57 destIP:2:0:0:0:0:0:0:985
nextHop:2:0:0:0:0:0:0:985
GGSN3(13842000): srcIP:3:0:0:0:0:0:2884:C8B0 destIP:3:0:0:0:0:0:0:986
nextHop:3:0:0:0:0:0:0:986
GGSN4(13842000): srcIP:4:0:0:0:0:0:2C8F:D70B destIP:4:0:0:0:0:0:0:987
nextHop:4:0:0:0:0:0:0:987
BS1(13843000): Packet delay srcAddress:1:0:0:0:0:0:206E:AC00
destAddress:1:0:0:0:0:0:0:984 delay(us):36800
BS2(13843000): Packet delay srcAddress:2:0:0:0:0:0:2479:BA57
destAddress:2:0:0:0:0:0:0:985 delay(us):36800
BS3(13843000): Packet delay srcAddress:3:0:0:0:0:0:2884:C8B0
destAddress:3:0:0:0:0:0:0:986 delay(us):36800
BS4(13843000): Packet delay srcAddress:4:0:0:0:0:0:2C8F:D70B
destAddress:4:0:0:0:0:0:0:987 delay(us):36800
M1(18264000): Received positive BindingAck from:1:0:0:0:0:0:206E:AC00
M2(18264000): Received positive BindingAck from:2:0:0:0:0:0:2479:BA57
M3(18264000): Received positive BindingAck from:3:0:0:0:0:0:2884:C8B0
M4(18264000): Received positive BindingAck from:4:0:0:0:0:0:2C8F:D70B
TCP1(99999999): TCP echo request:0
TCP3(99999999): TCP echo request:0
BS1(100000999): Packet delay srcAddress:1:0:0:0:0:0:0:984
destAddress:4:0:0:0:0:0:0:987 delay(us):836800
BS3(100000999): Packet delay srcAddress:3:0:0:0:0:0:0:986
destAddress:2:0:0:0:0:0:0:985 delay(us):836800
TCP1(144545130): TCP echo request:1
M1(144545130): Packet dropped due to wait queue overflow!
```

```
BS1(2187629277): Packet delay srcAddress:1:0:0:0:0:0:0:984
destAddress:4:0:0:0:0:0:0:987 delay(us):836800
BS4(2197923454): Packet delay srcAddress:4:0:0:0:0:0:2C8F:D70B
destAddress:FF02:0:0:0:0:0:0:1 delay(us):36800
TCP3(2208643353): TCP echo request:23
TCP3(2208643353): Packet 3 loss!
BS3(2208644353): Packet delay srcAddress:3:0:0:0:0:0:0:986
destAddress:2:0:0:0:0:0:0:985 delay(us):836800
TCP1(2270471718): TCP echo request:27
TCP1(2270471718): Packet 7 loss!
M1(2270471718): Packet dropped due to wait queue overflow!
GGSN1(2288050277): Address 4:0:0:0:0:0:0:987 Unreachable!
TCP1(2299485277): TCP echo request:28
TCP1(2299485277): Packet 8 loss!
BS1(2299486277): Packet delay srcAddress:1:0:0:0:0:0:0:984
destAddress:4:0:0:0:0:0:0:987 delay(us):836800
GGSN3(2309065353): Address 2:0:0:0:0:0:0:985 Unreachable!
TCP1(2319054579): TCP echo request:29
TCP1(2319054579): Packet 9 loss!
M1(2319054579): Packet dropped due to wait queue overflow!
BS3(2337437336): Packet delay srcAddress:3:0:0:0:0:0:2884:C8B0
destAddress:FF02:0:0:0:0:0:0:9 delay(us):76800
L3(2337437336): Packet delay srcAddress:F3:0:0:0:0:0:585:F098
destAddress:FF02:0:0:0:0:0:0:9 delay(us):77
R2(2337447336)->RIP: New Route
L5(2337448336): Packet delay srcAddress:F5:0:0:0:0:0:5F:4720
destAddress:FF02:0:0:0:0:0:0:9 delay(us):77
L3(2337448336): Packet delay srcAddress:F3:0:0:0:0:0:5F:32E0
destAddress:FF02:0:0:0:0:0:0:9 delay(us):77
L4(2337448336): Packet delay srcAddress:F4:0:0:0:0:0:5F:3D00
destAddress:FF02:0:0:0:0:0:0:9 delay(us):77
R1(2337458336)->RIP: New Route
GGSN4(2337458336)->RIP: New Route
L2(2337459336): Packet delay srcAddress:F2:0:0:0:0:0:5F:1F5A
destAddress:FF02:0:0:0:0:0:0:9 delay(us):77
L1(2337459336): Packet delay srcAddress:F1:0:0:0:0:0:5F:153B
destAddress:FF02:0:0:0:0:0:0:9 delay(us):77
L5(2337459336): Packet delay srcAddress:F5:0:0:0:0:0:5F:3DB7
destAddress:FF02:0:0:0:0:0:0:9 delay(us):77
BS4(2337459336): Packet delay srcAddress:4:0:0:0:0:0:2C8F:D70B
destAddress:FF02:0:0:0:0:0:0:9 delay(us):76800
L4(2337459336): Packet delay srcAddress:F4:0:0:0:0:0:17B:DAD8
destAddress:FF02:0:0:0:0:0:0:9 delay(us):77
GGSN2(2337469336)->RIP: New Route
GGSN1(2337469336)->RIP: New Route
BS2(2337470336): Packet delay srcAddress:2:0:0:0:0:0:2479:BA57
destAddress:FF02:0:0:0:0:0:0:9 delay(us):76800
L2(2337470336): Packet delay srcAddress:F2:0:0:0:0:0:990:656
destAddress:FF02:0:0:0:0:0:0:9 delay(us):77
L1(2337470336): Packet delay srcAddress:F1:0:0:0:0:0:D9A:1C12
destAddress:FF02:0:0:0:0:0:0:9 delay(us):77
TCP1(2337752210): TCP echo request:30
TCP1(2337752210): Packet 10 loss!
M1(2337752210): Packet dropped due to wait queue overflow!
TCP3(2351904795): TCP echo request:24
TCP3(2351904795): Packet 4 loss!
```

```
BS3(2351905795): Packet delay srcAddress:3:0:0:0:0:0:0:986
destAddress:2:0:0:0:0:0:0:985 delay(us):836800
```

Below is UMTS Core Network topology for **voice traffic** which consist user equipment, base station, CSCF, switch, router and TCP application.



UMTS Core Network Topology for Voice Traffic

Below are the list of step been logged during the **overall registration process simulation** according to UMTS Core Network topology for **voice traffic**.

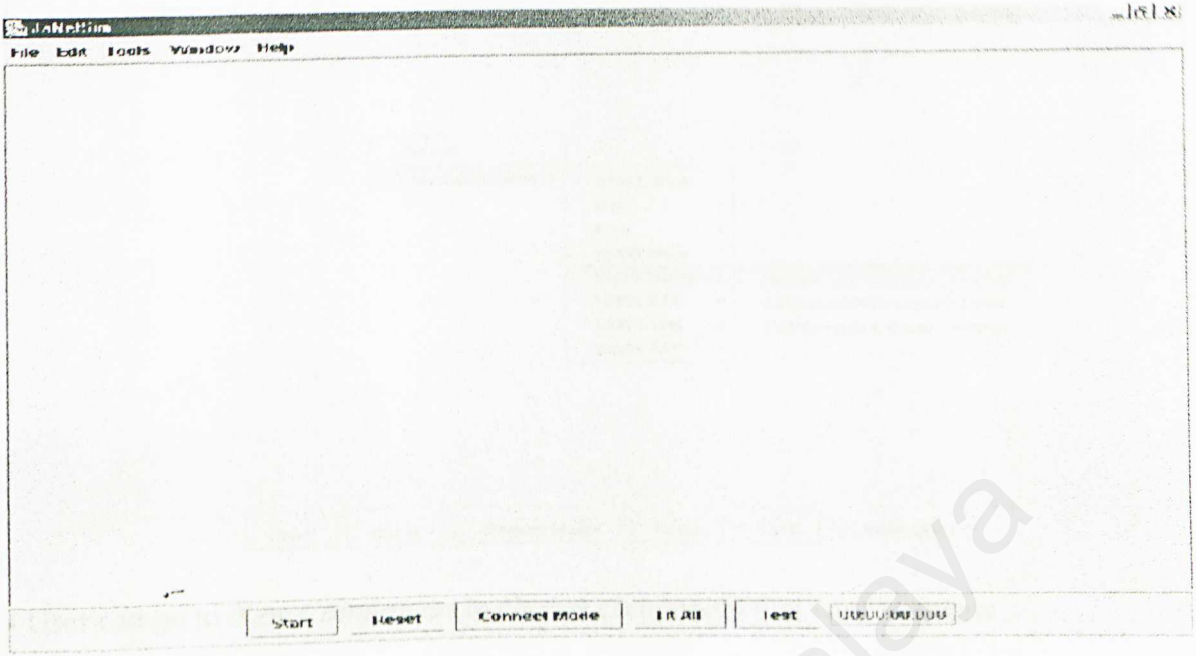
```
Hello, I am CSCF1
Hello, I am CSCF2
Hello, I am CSCF3
```

```
Hello, I am CSCF4
Hello, I am BS1
Hello, I am BS2
Hello, I am BS3
Hello, I am BS4
Hello, I am L1
Hello, I am L2
Hello, I am L3
Hello, I am L4
Hello, I am R1
Hello, I am R2
Hello, I am L5
Hello, I am TCP1
Hello, I am TCP2
Hello, I am TCP3
Hello, I am TCP4
Hello, I am M1
M1(0): Simulation Error - No outlink!
Hello, I am M2
M2(0): Simulation Error - No outlink!
Hello, I am M3
M3(0): Simulation Error - No outlink!
Hello, I am M4
M4(0): Simulation Error - No outlink!
M1(0): Connect to wirelesslink BS1
M2(0): Connect to wirelesslink BS2
M3(0): Connect to wirelesslink BS3
M4(0): Connect to wirelesslink BS4
BS1(1000): Packet delay srcAddress:0:0:0:0:0:0:0:0
destAddress:FF02:0:0:0:0:0:0:2 delay(us):36800
BS2(1000): Packet delay srcAddress:0:0:0:0:0:0:0:0
destAddress:FF02:0:0:0:0:0:0:2 delay(us):36800
BS3(1000): Packet delay srcAddress:0:0:0:0:0:0:0:0
destAddress:FF02:0:0:0:0:0:0:2 delay(us):36800
BS4(1000): Packet delay srcAddress:0:0:0:0:0:0:0:0
destAddress:FF02:0:0:0:0:0:0:2 delay(us):36800
BS1(4423000): Packet delay srcAddress:1:0:0:0:0:0:6C76:8400
destAddress:FF02:0:0:0:0:0:0:1 delay(us):36800
BS2(4423000): Packet delay srcAddress:2:0:0:0:0:0:689E:9C41
destAddress:FF02:0:0:0:0:0:0:1 delay(us):36800
BS3(4423000): Packet delay srcAddress:3:0:0:0:0:0:64C6:B480
destAddress:FF02:0:0:0:0:0:0:1 delay(us):36800
BS4(4423000): Packet delay srcAddress:4:0:0:0:0:0:60EE:CCBD
destAddress:FF02:0:0:0:0:0:0:1 delay(us):36800
M1(8844000): Send Location Register to home router
1:0:0:0:0:0:6C76:8400
M1(8844000): Creating Location Register srcAddress:1:0:0:0:0:0:0:984
destAddress:1:0:0:0:0:0:6C76:8400 HomeAddress:1:0:0:0:0:0:0:984
M2(8844000): Send Location Register to home router
2:0:0:0:0:0:689E:9C41
M2(8844000): Creating Location Register srcAddress:2:0:0:0:0:0:0:985
destAddress:2:0:0:0:0:0:689E:9C41 HomeAddress:2:0:0:0:0:0:0:985
M3(8844000): Send Location Register to home router
3:0:0:0:0:0:64C6:B480
M3(8844000): Creating Location Register srcAddress:3:0:0:0:0:0:0:986
destAddress:3:0:0:0:0:0:64C6:B480 HomeAddress:3:0:0:0:0:0:0:986
```

```
M4(8844000): Send Location Register to home router
4:0:0:0:0:0:60EE:CCBD
M4(8844000): Creating Location Register srcAddress:4:0:0:0:0:0:0:987
destAddress:4:0:0:0:0:0:60EE:CCBD HomeAddress:4:0:0:0:0:0:0:987
BS1(8845000): Packet delay srcAddress:1:0:0:0:0:0:0:984
destAddress:1:0:0:0:0:0:6C76:8400 delay(us):41600
BS2(8845000): Packet delay srcAddress:2:0:0:0:0:0:0:985
destAddress:2:0:0:0:0:0:689E:9C41 delay(us):41600
BS3(8845000): Packet delay srcAddress:3:0:0:0:0:0:0:986
destAddress:3:0:0:0:0:0:64C6:B480 delay(us):41600
BS4(8845000): Packet delay srcAddress:4:0:0:0:0:0:0:987
destAddress:4:0:0:0:0:0:60EE:CCBD delay(us):41600
CSCF1(13842000): Handling Location Register from COA:1:0:0:0:0:0:0:984
HomeAddress:1:0:0:0:0:0:0:984 expiry time:-1
CSCF1(13842000): Creating Binding Ack from:1:0:0:0:0:0:6C76:8400
to:1:0:0:0:0:0:0:984
CSCF2(13842000): Handling Location Register from COA:2:0:0:0:0:0:0:985
HomeAddress:2:0:0:0:0:0:0:985 expiry time:-1
CSCF2(13842000): Creating Binding Ack from:2:0:0:0:0:0:689E:9C41
to:2:0:0:0:0:0:0:985
CSCF3(13842000): Handling Location Register from COA:3:0:0:0:0:0:0:986
HomeAddress:3:0:0:0:0:0:0:986 expiry time:-1
CSCF3(13842000): Creating Binding Ack from:3:0:0:0:0:0:64C6:B480
to:3:0:0:0:0:0:0:986
CSCF4(13842000): Handling Location Register from COA:4:0:0:0:0:0:0:987
HomeAddress:4:0:0:0:0:0:0:987 expiry time:-1
CSCF4(13842000): Creating Binding Ack from:4:0:0:0:0:0:60EE:CCBD
to:4:0:0:0:0:0:0:987
CSCF1(13842000): srcIP:1:0:0:0:0:0:6C76:8400 destIP:1:0:0:0:0:0:0:984
nextHop:1:0:0:0:0:0:0:984
CSCF2(13842000): srcIP:2:0:0:0:0:0:689E:9C41 destIP:2:0:0:0:0:0:0:985
nextHop:2:0:0:0:0:0:0:985
CSCF3(13842000): srcIP:3:0:0:0:0:0:64C6:B480 destIP:3:0:0:0:0:0:0:986
nextHop:3:0:0:0:0:0:0:986
CSCF4(13842000): srcIP:4:0:0:0:0:0:60EE:CCBD destIP:4:0:0:0:0:0:0:987
nextHop:4:0:0:0:0:0:0:987
BS1(13843000): Packet delay srcAddress:1:0:0:0:0:0:6C76:8400
destAddress:1:0:0:0:0:0:0:984 delay(us):36800
BS2(13843000): Packet delay srcAddress:2:0:0:0:0:0:689E:9C41
destAddress:2:0:0:0:0:0:0:985 delay(us):36800
BS3(13843000): Packet delay srcAddress:3:0:0:0:0:0:64C6:B480
destAddress:3:0:0:0:0:0:0:986 delay(us):36800
BS4(13843000): Packet delay srcAddress:4:0:0:0:0:0:60EE:CCBD
destAddress:4:0:0:0:0:0:0:987 delay(us):36800
M1(18264000): Received positive BindingAck from:1:0:0:0:0:0:6C76:8400
M2(18264000): Received positive BindingAck from:2:0:0:0:0:0:689E:9C41
M3(18264000): Received positive BindingAck from:3:0:0:0:0:0:64C6:B480
M4(18264000): Received positive BindingAck from:4:0:0:0:0:0:60EE:CCBD
TCP1(99999999): TCP echo request:0
TCP3(99999999): TCP echo request:0
BS1(100000999): Packet delay srcAddress:1:0:0:0:0:0:0:984
destAddress:4:0:0:0:0:0:0:987 delay(us):836800
BS3(100000999): Packet delay srcAddress:3:0:0:0:0:0:0:986
destAddress:2:0:0:0:0:0:0:985 delay(us):836800
TCP3(197788888): TCP echo request:1
0:0:0:64C6:B480 destAddress:FF02:0:0:0:0:0:0:1 delay(us):36800
```

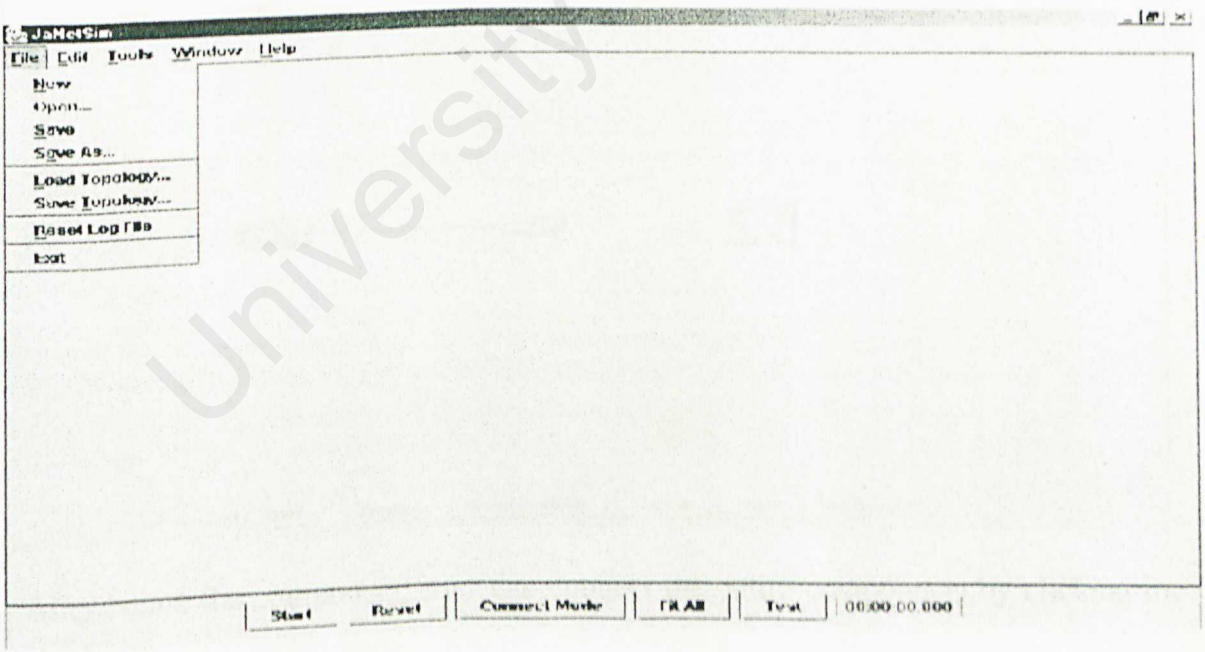
```
CSCF3(1963864353): Address 2:0:0:0:0:0:0:985 Unreachable!
TCP1(1974029392): TCP echo request:17
BS1(1974030392): Packet delay srcAddress:1:0:0:0:0:0:0:984
destAddress:4:0:0:0:0:0:0:987 delay(us):836800
BS2(2008739536): Packet delay srcAddress:2:0:0:0:0:0:689E:9C41
destAddress:FF02:0:0:0:0:0:0:9 delay(us):76800
L2(2008739536): Packet delay srcAddress:F2:0:0:0:0:0:15B3:411A
destAddress:FF02:0:0:0:0:0:0:9 delay(us):77
R1(2008749536)->RIP: New Route
L5(2008750536): Packet delay srcAddress:F5:0:0:0:0:0:5F:3DB7
destAddress:FF02:0:0:0:0:0:0:9 delay(us):77
L2(2008750536): Packet delay srcAddress:F2:0:0:0:0:0:5F:1F5A
destAddress:FF02:0:0:0:0:0:0:9 delay(us):77
L1(2008750536): Packet delay srcAddress:F1:0:0:0:0:0:5F:153B
destAddress:FF02:0:0:0:0:0:0:9 delay(us):77
R2(2008760536)->RIP: New Route
CSCF1(2008760536)->RIP: New Route
L4(2008761536): Packet delay srcAddress:F4:0:0:0:0:0:5F:3D00
destAddress:FF02:0:0:0:0:0:0:9 delay(us):77
L3(2008761536): Packet delay srcAddress:F3:0:0:0:0:0:5F:32E0
destAddress:FF02:0:0:0:0:0:0:9 delay(us):77
L5(2008761536): Packet delay srcAddress:F5:0:0:0:0:0:5F:4720
destAddress:FF02:0:0:0:0:0:0:9 delay(us):77
L1(2008761536): Packet delay srcAddress:F1:0:0:0:0:0:11DC:51F6
destAddress:FF02:0:0:0:0:0:0:9 delay(us):77
CSCF4(2008771536)->RIP: New Route
CSCF3(2008771536)->RIP: New Route
BS4(2008772536): Packet delay srcAddress:4:0:0:0:0:0:60EE:CCBD
destAddress:FF02:0:0:0:0:0:0:9 delay(us):76800
L4(2008772536): Packet delay srcAddress:F4:0:0:0:0:0:1D61:1F68
destAddress:FF02:0:0:0:0:0:0:9 delay(us):77
BS3(2008772536): Packet delay srcAddress:3:0:0:0:0:0:64C6:B480
destAddress:FF02:0:0:0:0:0:0:9 delay(us):76800
L3(2008772536): Packet delay srcAddress:F3:0:0:0:0:0:198A:3040
destAddress:FF02:0:0:0:0:0:0:9 delay(us):77
TCP3(2044805046): TCP echo request:15
BS3(2044806046): Packet delay srcAddress:3:0:0:0:0:0:0:986
destAddress:2:0:0:0:0:0:0:985 delay(us):836800
TCP1(2048845119): TCP echo request:18
M1(2048845119): Packet dropped due to wait queue overflow!
BS1(2057716442): Packet delay srcAddress:1:0:0:0:0:0:6C76:8400
destAddress:FF02:0:0:0:0:0:0:9 delay(us):76800
CSCF1(2074451392): Address 4:0:0:0:0:0:0:987 Unreachable!
L2(2082115125): Packet delay srcAddress:F2:0:0:0:0:0:15B3:411A
destAddress:FF02:0:0:0:0:0:0:1 delay(us):37
L4(2091613768): Packet delay srcAddress:F4:0:0:0:0:0:5F:3D00
destAddress:FF02:0:0:0:0:0:0:1 delay(us):37
L4(2124383832): Packet delay srcAddress:F4:0:0:0:0:0:1D61:1F68
destAddress:FF02:0:0:0:0:0:0:1 delay(us):37
CSCF3(2145227046): srcIP:3:0:0:0:0:0:0:986 destIP:2:0:0:0:0:0:0:985
nextHop:F3:0:0:0:0:0:5F:32E0
L3(2145228046): Packet delay srcAddress:3:0:0:0:0:0:0:986
destAddress:2:0:0:0:0:0:0:985 delay(us):837
R2(2145329246): srcIP:3:0:0:0:0:0:0:986 destIP:2:0:0:0:0:0:0:985
nextHop:F5:0:0:0:0:0:5F:3DB7
```

User Manual



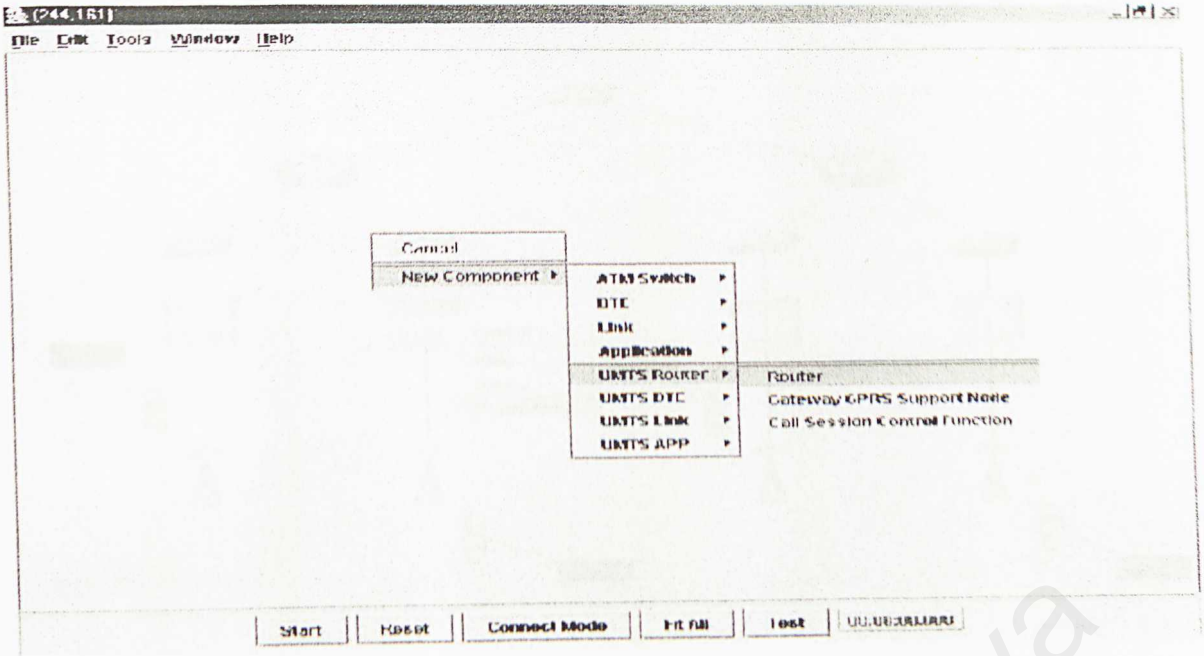
- This is the interface for JaNetSim

- It consists ***File, Edit, Tools, Window*** and ***Help***



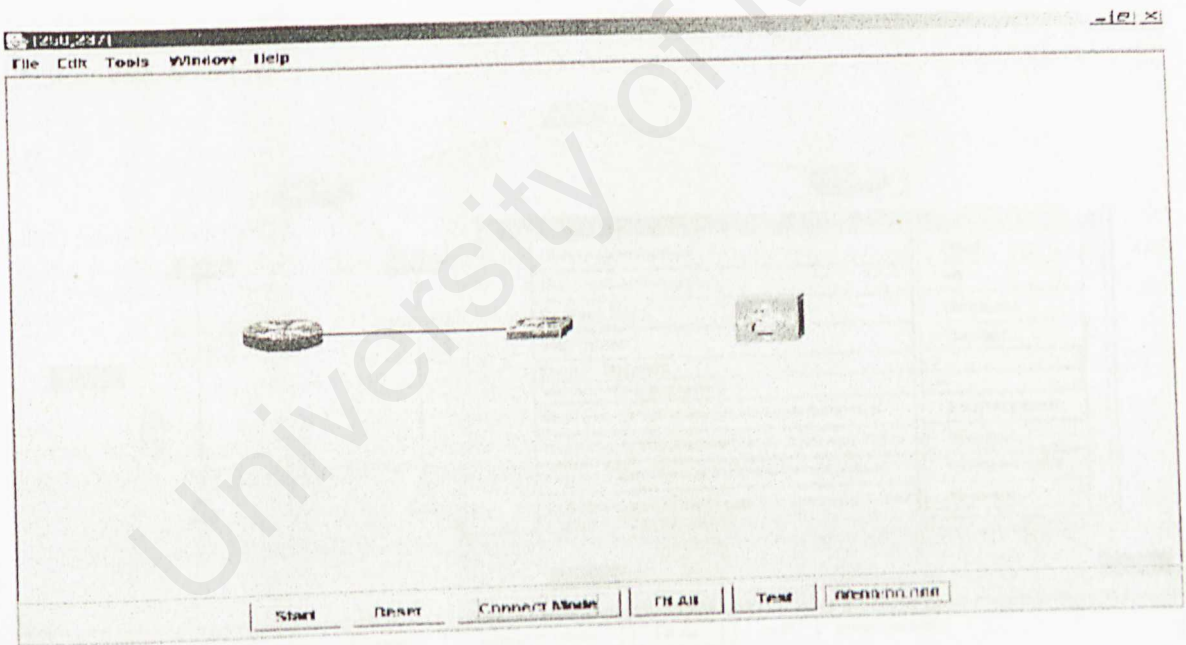
- User can ***Open*** or ***Load Topology*** if user already create topology before

- User can use ***New*** if want to create a new topology



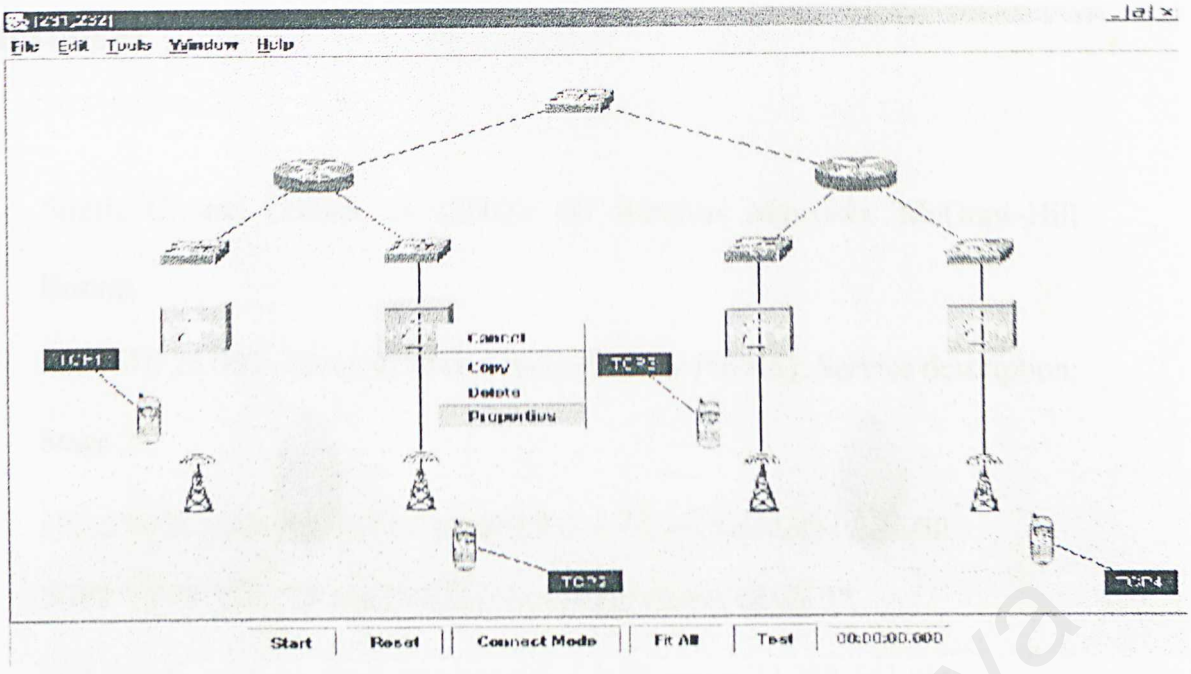
- User can go to ***New Component*** to choose component that want to create

- Example: ***New Component*** > ***UMTS Router*** > ***Router*** and click it

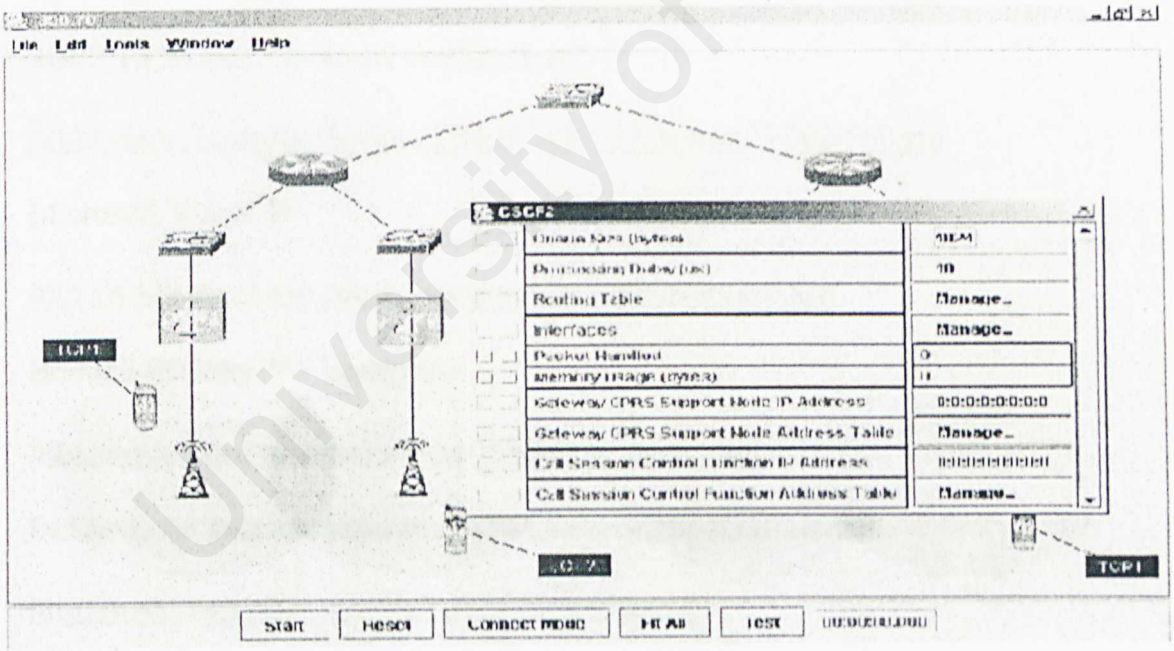


- After create the component, user can connect the entire components by clicking the

Connect Mode



- After connect all the components, user can go to configure the ***Properties*** by pointing the component and right click it



- From ***Properties***, user can set the parameter for each component

- After finish set all the components' properties, user can click ***Start*** to run the simulation and store the log file.

References

[1] Smith, C. and Collins, D. (2002). *3G Wireless Networks.* McGraw-Hill Boston.

[2] 3GPP TS 23.060: "General Packet Radio Service (GPRS); Service description; Stage 2".
http://www.3gpp.org/ftp/Specs/latest/Rel-5/23_series/23.060-520.zip

[3] 3GPP TS 23.228: "IP Multimedia Subsystem (IMS); Stage 2".
http://www.3gpp.org/ftp/Specs/latest/Rel-5/23_series/23.228-541.zip

[4] 3GPP TS 24.228: "Signalling flows for the IP multimedia call control based on SIP and SDP".
http://www.3gpp.org/ftp/Specs/latest/Rel-5/24_series/24.228-500.zip

[5] 3GPP TS 23.002: "Network Architecture".
http://www.3gpp.org/ftp/Specs/latest/Rel-5/23_series/23.002-560.zip

[6] Microsoft Visual J#
http://msdn.microsoft.com/vjsharp/productinfo/overview.asp

[7] Borland JBuilder™ 7 Enterprise
http://www.borland.com/jbuilder/enterprise/index.html

[8] INSANE An Internet Simulated ATM Networking Environment.
http://www.ca.sandia.gov/~bmah/Software/Insane

[9] The REAL Network Simulator
http://minnie.cs.adfa.edu.au/REAL/index.html

[10] Java™ 2 SDK, Standard Edition, version 1.3.1 Summary of News Features and Enhancements. *http://www.java.sun.com.* Sun Microsystems, Inc.

[11] Nada G., el. (1998). The NIST ATM/HFC Network Simulator Operation and Programming Guide Version 4.0. National Institute of Standards and Technology.

[12] JaNetSim Simulator
http://2.2.185.109.181/janetsim

[13] Deitel & Deitel. (2002). *JAVA How to Program,* Fourth edition. Prentice Hall Inc. New Jersey.

[14] Object Oriented Basic Concepts and Advantages.
http://www.mmrg.ecs.soton.ac.uk/publications/archive/melly1995a/html/node3.html. M.Melly.